LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

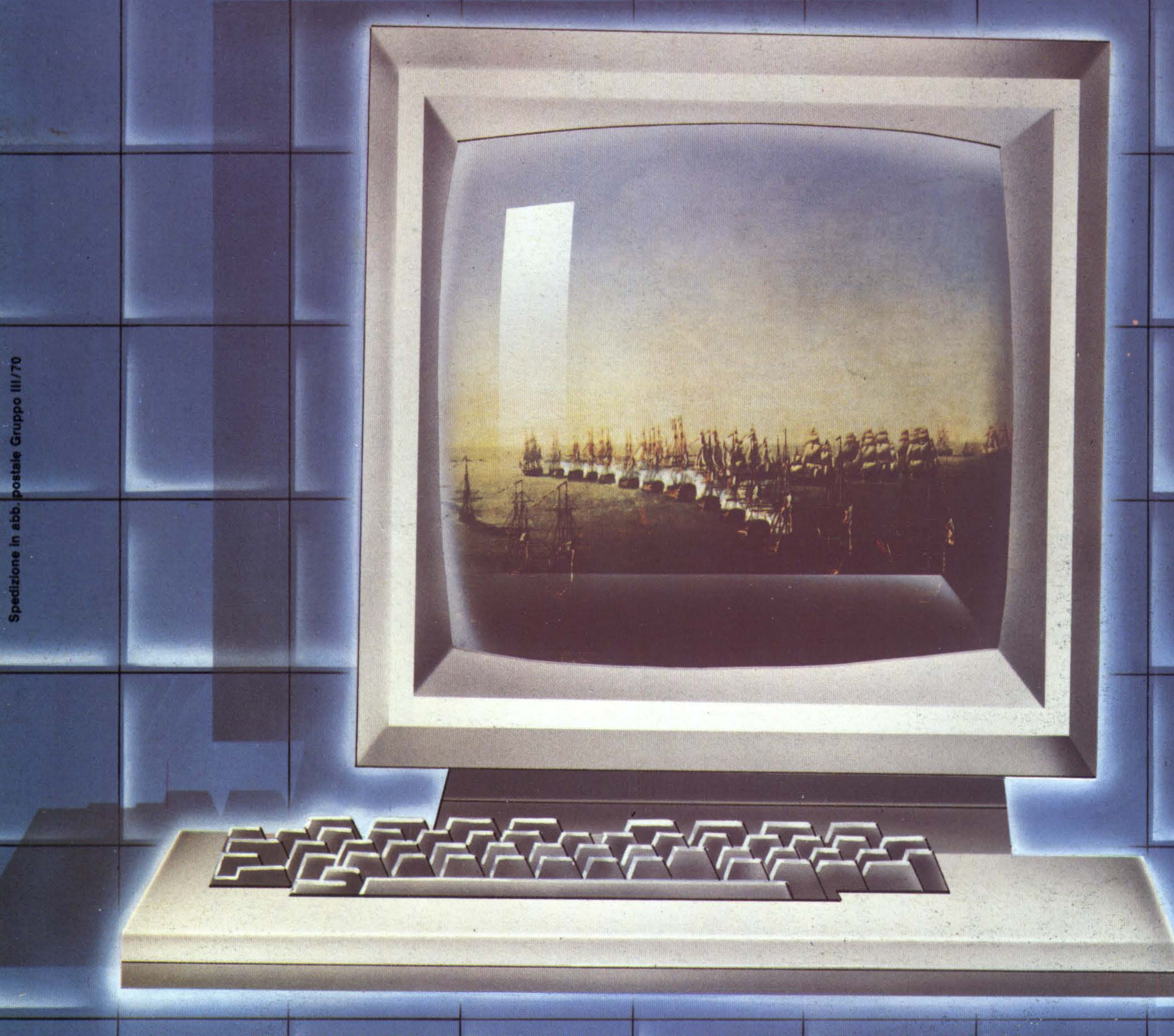
PERSONAL SOFTWARE

ANNO 3 N. 15
FEBBRAIO 1984 - L. 3.500

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Copia riservata agli abbonati

Spedizione in abb. postale Gruppo III/70



- UN POTENTE WORD PROCESSOR PER DAI
- IMPARIAMO IL LINGUAGGIO MACCHINA CON IL VIC E C 64
- LINGUAGGIO MACCHINA PER SINCLAIR
- DAL BASIC AL PASCAL
- ABUKIR 1798: SIMULAZIONE DI UNA BATTAGLIA NAVALE CON IL PET CBM

MILANO 22-26 MAGGIO 1984

VIDEO GAMES USA

PERCHÈ UNA NUOVA DATA?

Per una ragione più che valida, VIDEO GAMES USA entra a far parte di BIT USA, la prestigiosa mostra di home e personal computer americani. E l'edizione '84, arricchita dalla presenza dei videogiochi, sarà più interessante che mai!

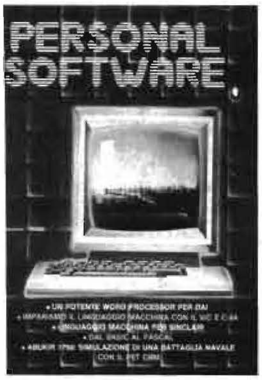
NON DIMENTICATE DUNQUE di visitare la sezione Videogiochi di BIT USA 84, dal 22 al 26 maggio, presso il Centro Commerciale Americano.

USA

CENTRO COMMERCIALE AMERICANO

Via Gattamelata 5, 20149 Milano
Tel. (02) 46.96.451 Telex 330208 USIMC-I

La mostra è realizzata in collaborazione con le riviste del **Gruppo Editoriale Jackson.**



In copertina: riproduzione di un dipinto storico raffigurante la battaglia di Abukir del 1798, oggetto dell'articolo di G.U. Barzaghi per il PET CBM.

ARTICOLI | GUIDA

RUBRICHE

PERSONAL SOFTWARE
N. 15
FEBBRAIO 1984



OGGI
ANTICIPAZIONI
SUL FUTURO
SIOA
salone
dell'informatica,
della telematica
e della
organizzazione
aziendale
Bologna,
25-29 febbraio 1984
quartiere fieristico
Informatica
Telecomunicazioni
Telematica
Servizi di consulenza
ed assistenza
alle imprese
Attrezzature
per l'ufficio
promosso da:
ANIE - Associazione Nazionale
Industrie Elettrotecniche
ed Elettroniche
Ente Autonomo
per le Fiere di Bologna
Fondazione Guglielmo Marconi
IRSAC - Istituto di Studi
e Ricerche sulle Attività
Commerciali e Produttive
Gestione:
GE.MA. General Management S.r.l.
Direzione Operativa:
Bologna, via de' Buttieri, 7/2A
Tel. 051/308952 - 340882
Telex 510878
Studio Moruzzi - Bologna

EDITORIALE

Alla ricerca del regalo intelligente

di Riccardo Paolillo

Non è un mistero per nessuno che quella che doveva essere una frenetica corsa agli acquisti in occasione delle recenti festività natalizie, si è dimostrata in realtà una lenta e ponderata camminata.
La crisi delle vendite ha colpito, secondo quanto riferiscono gli operatori commerciali, un po' tutti i settori non ultimo, ovviamente, quello dell'elettronica civile (TV color, HI-FI, ecc.).
Ma in questo panorama così nero, che rispecchia un difficile momento economico, spicca un fatto nuovo e da molti inatteso che, se in termini generali non modifica una tendenza altamente negativa, costituisce senza dubbio materia per qualche osservazione: il boom nelle vendite di home computer.
Questa sì che è stata una corsa all'acquisto che ha stupito tutti, in particolare gli stessi commercianti che si sono trovati in molti casi e con ovvio disappunto, impossibilitati a far fronte alla marea di richieste ricevute per i più popolari personal.
Bersagliato da una pubblicità spesso martellante, condizionato dalle varie guide ai regali intelligenti pubblicate da settimanali e quotidiani, il compratore italiano ha "scoperto" il calcolatore personale quello che "costa circa come un videogame, ma vuol mettere, lei si mette in casa un vero computer".
La piccola cronaca delle vendite natalizie ci ha riservato episodi assolutamente inediti e per certi versi incredibili. Così ci sono stati negozi che hanno organizzato vere e proprie liste di attesa vendendo materiale che avrebbero consegnato all'acquirente anche più di un mese dopo; negozianti a loro volta costretti a lunghe code presso i distributori delle varie case per contendersi i pochi (rispetto alla domanda) calcolatori disponibili.
Un mio conoscente, capitato in un negozio in cui era stata organizzata una catena di montaggio della vendita (qui i soldi, qua il calcolatore, grazie e arrivederci) ha avuto l'ardire di richiedere un depliant esplicativo di uno dei personal più diffusi: il commesso l'ha immediatamente liquidato dicendogli di non distrarlo visto che stava lavorando.
Comunque sia, tutto ciò ha contribuito a far lievitare il numero di utilizzatori o comunque di possessori di home computer.
E proprio perché, svanita l'euforia della novità, il personal non diventi un semplice giocattolo che non diverte più, ci permettiamo di dare qualche modesto consiglio ai nuovi utilizzatori.
Innanzitutto le basi: fare il programmatore è per molti un mestiere e come tale si impara, facilmente, ma non si improvvisa. Imparare a programmare in BASIC è semplice e alla portata di tutti, ma non si può pretendere di farlo dall'oggi al domani.
Diciamo queste cose perché, a volte, pubblicità poco veritiere e operatori scarsamente professionali, illudono in modo scorretto l'ignaro acquirente.
In ogni caso esistono parecchi testi in italiano (molti editi dal Gruppo Editoriale Jackson, citando a caso (?!!), che ha in catalogo parecchi volumi destinati proprio ai neofiti) disponibili presso le migliori librerie, come pure ci sono riviste specializzate (una è ovviamente **Personal Software**) che si dedicano in particolare a home e personal computer.
Riteniamo che questi semplici strumenti, uniti ad una buona dose di pazienza ed interessamento personale, siano ampiamente sufficienti a quanti vogliono addentrarsi nell'affascinante mondo dell'informatica.
In breve tempo si sarà in grado di scrivere simpatici programmi, come semplici videogame o piccole applicazioni domestiche (conto bancario personale, agenda telefonica, ricette di cucina) che se indubbiamente non muteranno di molto la qualità della vita familiare, forniranno vantaggi indiretti probabilmente maggiori: la programmazione ci fornisce un metodo per pensare meglio, in modo più logico e razionale e questo, sicuramente, non è poco. ■



POSTA

A proposito di collaborazioni

Ho 13 anni, posseggo un computer Commodore CBM 64 e ho costruito un videogame. Si tratta di un'astronave che deve atterrare in 3 pianeti. Vorrei sapere da voi se lo posso vendere, ed eventualmente a chi. Lo posso vendere a voi? È un bel gioco, ci ho messo un anno per farlo. Voi prendete list del Commodore 64?

P.S. anche un mio amico ha inventato un gioco.

Igor Schiaroli
Roma

Fra le tante lettere che ci propongono offerte di collaborazione abbiamo scelto questa perché è sicuramente la più carina e spontanea. Rispondiamo al simpatico Igor cercando di chiarire, anche a beneficio dei molti altri che ci hanno scritto, la nostra posizione a proposito di questo argomento.

Personal Software, *come è noto, si basa principalmente sui contributi dei lettori e quindi i vostri lavori sono graditissimi e attentamente valutati.*

Tenete conto, però, che noi non acquistiamo software dai lettori, ma riconosciamo un compenso (intorno alle 50.000 lire nette per ogni pagina di rivista) per tutti i lavori pubblicati.

In ogni caso ci impegnamo a non sfruttare commercialmente o divulgare il materiale non ritenuto idoneo per la pubblicazione.

In questi giorni stiamo continuando a spedire la Guida agli autori (è un fascicoletto di 16 pagine) a tutti i lettori che l'hanno richiesto; dovete avere un po' di pazienza perché le richieste sono state numerosissime. A questo proposito rinnoviamo l'invito a richiedere la guida solo se veramente interessati a collaborare, per poter effettuare le spedizioni in tempi brevi.

Infine un consiglio sottovoce: generalmente gli articoli dattiloscritti in buon italiano e con i programmi memorizzati su cassetta o dischetto hanno la precedenza...

Didattica e computer

Sono un ragazzo appassionato di computer ed assiduo lettore della vostra rivista (tanto appassionato che mi sono abbonato per non perdere un numero) che trovo interessante ed utile.

Io posseggo un VIC 20 non espanso, mi diverto a fare giochi o programmi di altra utilità, ed ogni tanto mi dedico anche a conversioni; da un po' di tempo mi sono dedicato a problemi matematici e scientifici, ma non ho potuto trovare finora un programma che faccia operazioni e successivi grafici sui numeri complessi. Ciò mi serve perché sono uno studente, frequento il 5° anno di Elettronica Industriale a Messina, a scuola ci insegnano i linguaggi BASIC e FORTRAN, e mi è quasi indispensabile per la risoluzione immediata di esercizi o la costruzione di grafici di funzioni complesse.

Sarei quindi veramente lieto di trovare questo programma su un prossimo numero di **Personal Software.**

Domenico Bonaccorso
Messina

Già da un po' di tempo stiamo pensando di pubblicare alcuni articoli che sviluppino argomenti connessi ad un utilizzo nella didattica del personal. Ci ripromettiamo di pubblicare anche programmi inerenti applicazioni matematiche e magari anche sugli argomenti specifici da lei proposti.

In ogni caso invitiamo i lettori a contribuire sia esponendo eventuali richieste, sia mediante l'invio di articoli sull'argomento. Saranno particolarmente graditi i consigli e gli interventi degli insegnanti che hanno introdotto (o pensano di farlo) il calcolatore in classe.

Dal VIC 20 al C64

Possiedo da qualche tempo un Commodore 64 e precedentemente ho avuto per circa 6/7 mesi un VIC 20.

Ho ancora il registratore che avevo acquistato per il VIC ed ultimamente ho acquistato il floppy singolo 1541 per il 64. Vorrei sapere (mi hanno assicurato che è possibile) come posso trasferire; anche solo per il LIST, un programma registrato su cassetta con il VIC, dal nastro alla memoria del 64. Ciò mi permetterebbe di recuperare almeno parte dei molti programmi che ho in versione VIC 20.

Mi piacerebbe anche che pubblicaste uno o più articoli sulla gestione dei file con il 64 ed il disco.

Inoltre perché non pubblicate qualcosa sulle differenze esistenti tra i vari BASIC Commodore, dal VIC in su?

Ed ancora, non potreste aggiungere ai listati VIC e CBM le modifiche necessarie a farli girare sul 64 (a parte la differente tabulazione e le ovvietà...)?

Forse le mie richieste sono troppe, ma possono interessare molti possessori di 64 arrivati a questa macchina dopo il VIC. E forse questa può essere una forma (comoda..!) di collaborazione alla rivista.

Alex Conte
Torino

Il suo problema è comune a molti altri utenti di VIC 20 convertiti al C 64. Recentemente abbiamo cominciato ad occuparci della compatibilità tra le due macchine facendo delle prove in Redazione.

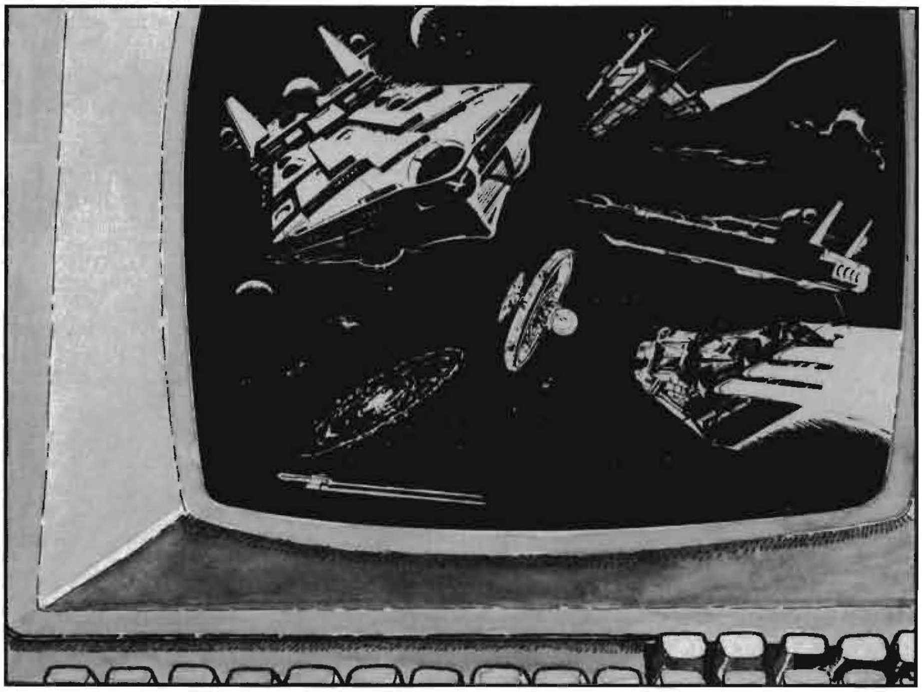
Purtroppo, nonostante venga usato lo stesso registratore e le routine di I/O siano simili, non esiste compatibilità diretta in quanto i programmi vengono caricati in aree diverse di memoria e quindi variano anche i puntatori alle variabili.

Per sfruttare in modo automatico i programmi scritti per VIC 20 occorrerebbe realizzare un rilocatore di programmi BASIC che trasferisca il programma e i relativi puntatori nelle locazioni di memoria previste dal sistema operativo del C 64.

Dato che questo programma, ovviamente in linguaggio macchina, non è semplicissimo da realizzare, si preferisce normalmente ribattere il listato. Tra l'altro, agendo in modo manuale, si può modificare direttamente per quanto riguarda le parti relative alla visualizzazione di caratteri, alla grafica e al suono, cosa che risulterebbe ovviamente impossibile effettuare in modo automatico.

Comunque di questo argomento ci ripromettiamo di parlare in un articolo di imminente pubblicazione. Presto parleremo anche della gestione del disco, dato che questa periferica si segnala per un rapporto qualità prezzo veramente notevole ed è facile prevederne una veloce diffusione.

A proposito di conversioni, infine, abbiamo verificato che esiste compatibilità tra il C 64 e i vecchi PET e CBM.



Linguaggio macchina per Sinclair

— Parte seconda —

Dove e perché sistemare i codici: una routine per Spectrum e ZX81 e un po' di teoria non fa mai male

di *Bruno Del Medico*

La prima parte di questo articolo, pubblicata sul numero 12 di **Personal Software**, era dedicata esclusivamente allo ZX81. La grande diffusione raggiunta dallo Spectrum ci ha indotto a estendere la trattazione anche a questo personal.

In questo numero riprenderemo la discussione dei metodi di immagazzinamento delle routine LM, con particolare riguardo allo Spectrum.

Successivamente il discorso Spectrum-ZX81 potrà essere unificato perché entrambi i sistemi usano lo stesso microprocessore, lo Z80.

I possessori di ZX81 scuseranno qualche inevitabile ripetizione.

Consiglio comunque di non trascurare la parte finale, importantissima, dedicata al concetto di CARRY o riporto.

Il programma PROVA LM/1

Inserite il programma BASIC del listato 1 (listato 2 per lo ZX81), ma aspettate a farlo girare.

Questo programma si può dividere in 3 parti:

a) linee 10-30,
b) linea 40,
c) linee 50-70.

La prima routine è un ciclo FOR K che va da 1 a 5. Ogni volta che il ciclo viene eseguito il computer scrive qualcosa nella posizione K, K.

In effetti, nel caso dell ZX81, scrive un carattere alfabetico scelto a caso tra i 26 disponibili, perché l'espressione:

INT (RND ★ 25) + 38

restituisce un numero a caso compreso tra 38 e 63, e la funzione CHR$ usata in relazione ai numeri da 38 a 63 dà sempre un carattere alfabetico.

Al termine di questa prima routine l'output che appare sullo schermo è simile a quello illustrato nella figura 1.

Per lo Spectrum appariranno dei quadratini colorati, al posto dei caratteri alfabetici.

La linea 40 ferma il programma per due secondi, per consentire di oservare il primo quadro video.

La terza routine (linee 50-70) è simile alla prima; il computer esegue un ciclo che va da 1 a 5, e ogni volta scrive qualche cosa nella posizione I, K.

Anche chi non ha seguito la prima parte dell'articolo può facilmente intuire che USR 32000 e USR 17200 rappresentano un numero, una lettera o una stringa, qualcosa insomma che possa essere stampato sullo schermo.

USR 32000 è sempre un numero. È il risultato dei calcoli eseguiti in una routine memorizzata a partire dall'indirizzo 32000.

La routine potrebbe essere memorizzata a partire da qualsiasi altro indirizzo che non cambierebbe niente, per esempio per lo ZX81 usiamo: USR 17200.

Volendo stabilire una analogia con il BASIC, l'istruzione PRINT

```
  1 REM PROVA LM/1 - SPECTRUM

 10 FOR K=1 TO 5
 20 LET A=INT (RND*8): PRINT AT
K,K; INK A;"■"
 30 NEXT K
 40 PAUSE 100
 50 FOR K=1 TO 5
 60 PRINT AT K,K;USR 32000
 70 NEXT K


  1 REM PROVA LM/1 - ZX81

 10 FOR K=1 TO 5
 20 PRINT AT K,K;CHR$ (INT (RND
*25)+33)
 30 NEXT K
 40 PAUSE 100
 50 FOR K=1 TO 5
 60 PRINT AT K,K;USR 17200
 70 NEXT K
```

Listati 1 e 2. *Esempio di programma che utilizza una routine LM.*

```
    1 REM SOUBROUTINE BASIC
          EQUIVALENTE ALLA
          ROUTINE LM CARICATA
          SULLO SPECTRUM

32000 LET HL=PEEK 23664+256 *
      PEEK 23665
32003 LET BC=6144
32006 LET HL=HL+BC
32008 LET C=PEEK HL
32010 RETURN


    1 REM SOUBROUTINE BASIC
          EQUIVALENTE ALLA
          ROUTINE LM CARICATA
          SULLO ZX81

17200 LET HL=PEEK 16398+256 *
      PEEK 16399
17203 LET C=PEEK HL
17206 RETURN
```

Listati 3 e 4. *Le routine BASIC di questi listati svolgono lo stesso compito della routine LM descritta nell'articolo, occupando circa 70 byte nel caso dello ZX81 e oltre 100 nel caso dello Spectrum. Invece la routine LM equivalente (tabella 1) occupa 7 byte nel caso dello ZX81 e 11 nel caso dello Spectrum.*

USR 17200 può essere considerata equivalente a:

GOSUB	17200
PRINT	(il risultato della elaborazione avvenuta nella subroutine 17200)

Naturalmente però 17200 è riferito ad un indirizzo della memoria RAM e non ad una linea di programma.

Nello ZX81 senza espansione di memoria la RAM va dall'indirizzo

16384 al 17408. Scrivendo la routine dall'indirizzo 17200 in poi si viene a trovare abbastanza lontana dal programma BASIC, precauzione opportuna per evitare sovrapposizioni. Analogamente nello Spectrum 16 Kbyte, la memoria RAM va dall'indirizzo 16384 all'indirizzo 32599; scegliamo l'indirizzo 32000 perché è abbastanza alto (ricordiamoci CLEAR 31999, vedi dopo).

Tornando al programma, manca ancora la routine in linguaggio macchina, infatti agli indirizzi destinati a tale routine non c'è niente. Per verificarlo scrivete:

ZX81	*Spectrum*
PRINT PEEK 17200	PRINT PEEK 32000

l'istruzione PEEK restituisce il contenuto dell'indirizzo indicato. Premendo NEWLINE o ENTER sullo schermo appare: 0.

Continuando con gli indirizzi successivi si ottiene sempre 0, perché quella parte di memoria RAM non è occupata da niente; possiamo quindi sistemarci la nostra routine LM. Per il momento ci basta sapere che, è composta da una serie di numeri decimali. Per lo Spectrum:
42, 132, 92, 1, 0, 24, 9, 78, 6, 0, 201
Per lo ZX81:
42, 14, 64, 78, 6, 0, 201

Il metodo di più immediata comprensione per immagazzinare la routine è il seguente:

ZX81	*Spectrum*
POKE 17200, 42	POKE 32000, 42
POKE 17201, 14	POKE 32001, 132
...	...
POKE 17206, 201	POKE 32010, 201

Battendo queste istruzioni si carica in memoria la routine LM, che per lo Spectrum occupa gli indirizzi dal 32000 al 32010. Poiché ogni indirizzo è un byte, la routine relativa allo Spectrum occupa il byte, quella relativa allo ZX81 ne occupa 7.

Ora che anche la routine LM è sistemata in memoria, date il RUN per verificare l'effetto della linea 60 (listati 1 e 2).

Sullo schermo appare un output simile a quello della figura 1.

Solo ZX81: dopo una pausa di due secondi, appare un nuovo output simile a quello della figura 2.

I numeri del secondo output rappresentano i codici dei caratteri scritti nel primo. Infatti:

CODE C = 40
CODE Z = 63 e così via.

Il computer ha percorso un certo tracciato, scrivendo sullo schermo una serie di lettere dell'alfabeto. Successivamente ha ripercorso lo stesso tracciato, e per ogni punto:
a) ha letto il carattere scritto in quel punto;
b) ha calcolato il codice del carattere;
c) ha scritto nello stesso punto il codice calcolato.

Solo Spectrum: ognuno dei numeri che appaiono nel secondo output rappresenta il valore della funzione ATTR riferito a quel punto dello schermo.

Il computer ha percorso un certo tracciato disegnando cinque quadratini colorati. Successivamente lo ha ripercorso e per ogni punto:
a) ha riconosciuto il colore disegnato in quel punto;
b) ha calcolato il relativo valore della funzione ATTR;
c) ha scritto il valore di ATTR nello stesso punto.

In entrambi i casi (ZX81 e Spectrum) le operazioni *a* e *b* sono state eseguite dalla routine LM.

Volendo continuare l'analogia

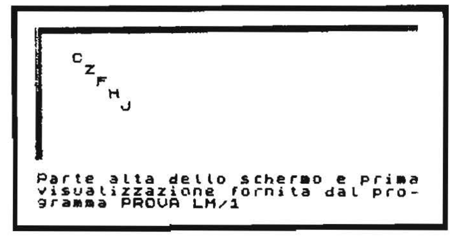


Figura 1. *ZX81: Prima visualizzazione offerta dal programma PROVA LM/1.*
Spectrum: I caratteri diventano dei quadratini colorati.

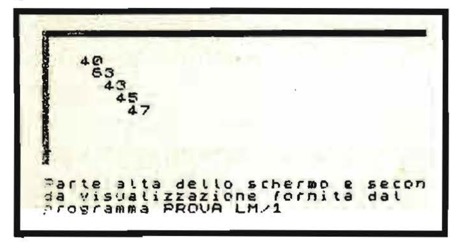


Figura 2. *Seconda visualizzazione offerta dal programma PROVA LM/1. La routine LM legge i caratteri presenti sul video e scrive al loro posto il codice equivalente.*

con il BASIC possiamo dire che la linea 60 e la routine LM si comportano come illustrato nella figura 3. Volendo ottenere un risultato uguale con una subroutine in BASIC si potrebbe usare quella illustrate nei listati 3 (Spectrum) e 4 (ZX81).

Solo ZX81: i byte 16398 e 16399 contengono un indirizzo di memoria che viene caricato nella variabile numerica HL. Con l'istruzione PEEK HL si legge il contenuto di questo indirizzo. Esso equivale al numero di codice del carattere presente nel punto dello schermo che sta per essere scritto, cioè al codice della posizione di stampa.

Solo Spectrum: i byte 23684 e 23685 contengono un indirizzo di memoria che viene caricato in HL. Si aggiunge ad HL il valore 6144, perché l'indirizzo che vogliamo leggere si trova 6144 byte più in alto

nella memoria. Con l'istruzione LET C = PEEK HL si legge il contenuto di questo indirizzo: esso equivale al valore degli ATTRIBUTI relativi al punto dello schermo che sta per essere scritto. Il valore degli attributi varia a seconda della diversa colorazione di quel punto.

Occorre notare che mentre la routine LM è lunga una decina di byte i listati 3 e 4 usano una quantità di memoria enormemente superiore per svolgere il medesimo compito. Già la prima linea ne occupa molti di più.

Di tutti i concetti esposti fino qui, è necessario avere ben compreso almeno questi:
— con l'istruzione POKE possiamo scrivere delle istruzioni (sotto forma di numeri decimali) in determinati indirizzi della memoria RAM, partendo da un indirizzo qualsiasi purché separato dall'area del programma (per evitare confusioni);
— con l'istruzione PEEK possiamo rileggere le istruzioni scritte;
— con l'istruzione USR le mandiamo in esecuzione.

Facciamo ora alcune considerazioni sul programma PROVA LM/1 (listati 1 e 2) appena sperimentato:
— usando il RUN la routine LM non viene cancellata;
— usando il LIST non è possibile vedere la routine LM; l'unico metodo possibile per listarla consiste in una noiosa ripetizione di istruzioni PEEK;
— la routine LM non viene registrata su nastro con il SAVE.

Registrate il programma PROVA LM/1 con la routine LM caricata.

Riavvolgete poi il nastro ed eseguite il LOAD. Noterete che la routine LM è ancora presente e il programma gira. Si tratta però di una illusione: la routine è stata semplicemente ignorata sia in fase di SAVE che in fase di LOAD. Spegnendo il computer e riprovando il LOAD, si perde la routine LM.

Ricapitolando: possiamo sistemare una routine LM in qualsiasi parte alta della memoria RAM, ma con questo metodo la routine non è né listabile né registrabile su nastro (sullo Spectrum c'è un'altra possibilità, vedi dopo).

Immagazzinamento della routine LM nella prima linea di programma

C'è un altro metodo che consente di immagazzinare una routine LM in una posizione diversa della memoria RAM, in modo tale che sia possibile il SAVE insieme al programma BASIC ed una certa forma di LIST (non sempre affidabile).

Nel numero 12 di **Personal Software** abbiamo visto che con lo ZX81 si può scrivere all'inizio del programma BASIC un pezzo di programma fasullo, sul quale poi sovrapponiamo i caratteri della routine LM. Il programma BASIC utile, viene scritto in coda a questa prima parte.

Ovviamente, se la nostra routine LM deve essere composta da 7 byte, allora il programma BASIC inutile deve essere composto anch'esso da ALMENO 7 byte: se poi sono 8 o 10 si spreca soltanto qualche byte di memoria.

Un pezzo di programma BASIC inutile composto da 7 byte è:

1 REM 0000000

Se la routine fosse stata composta da 15 byte, avremmo usato una linea REM composta da 15 zeri. Però è difficile e noioso contare 15 zeri. Allora avremmo potuto scrivere:

1 REM 123456789012345

In effetti non ha importanza quale tipo di carattere sia contenuto nella linea REM: è importante solo che la quantità di caratteri sia ALMENO PARI alla lunghezza in byte della routine LM.

Anche con lo Spectrum la routine LM, può essere sistemata nella parte bassa della memoria, compresa tra l'indirizzo 16384 e l'ultimo byte occupato; in questo modo viene memorizzata automaticamente insieme al programma BASIC. (Vedremo poi come nello Spectrum sia possibile e preferibile sistemare la routine nella parte alta).

Alcune aree dall'indirizzo 16384 in poi, pur facendo parte della memoria RAM, non possono essere utilizzate perché sono destinate a contenere altre informazioni. In particolare non possono essere usati gli indirizzi dal 16384 al 23734.

Anche una certa quantità di indirizzi successiva al 23734 non può essere usata: l'area del programma comincia subito dopo questa certa quantità di locazioni, che purtroppo è variabile.

Quindi il primo byte dell'area del programma non ha un indirizzo fisso di partenza nello Spectrum.

L'indirizzo può variare, ma il computer lo conserva sempre aggiornato nella sua memoria, e si ottiene con questa istruzione:

PRINT PEEK 23635 + 256 ★ PEEK 23636

Se volessimo immagazzinare la routine nei primi byte della memoria RAM riservata al programma, come con lo ZX81, dovremmo scrivere:

POKE (PEEK 23635 + 256 ★ PEEK 23636), 42
POKE (PEEK 23635 + 256 ★ PEEK 23636) + 1,92
...
POKE (PEEK 23635 + 256 ★ PEEK 23636) + 10, 201

Però abbiamo già visto nel caso dello ZX81 che non è possibile sistemare così la routine LM, perché i numeri scritti con l'istruzione POKE andrebbero a sovrapporsi alle prime istruzioni del programma BASIC cancellandole.

Ricorriamo anche nel caso dello Spectrum alla soluzione prospettata per lo ZX81: scriviamo un pezzo di programma BASIC inutile, che vada ad occupare le prime locazioni di memoria nell'area del programma. In pratica basta scrivere la linea:

1 REM 12345678901
seguita dal programma, con linee numerate da 2 in poi.

Gli indirizzi occupati dai caratteri della REM della linea 1 sono destinati alle istruzioni della routine LM.

Quindi la REM deve contenere almeno tanti caratteri quanti sono i byte della routine LM.

Il calcolo relativo ai byte occupati dalla linea 1 può essere effettuato con l'aiuto della tabella 1 (pubblicata nella prima parte, **Personal Software** n. 12) che è valida anche per lo Spectrum. La linea contiene 11 caratteri e occupa 17 byte: undici per i caratteri, 5 per la linea e uno per il REM.

Volendo inserire la routine LM nella linea 1 a partire dal primo carattere della REM occorre usare come indirizzo di partenza:

PEEK 23635 + 256 ★
PEEK 23636 + 5

A questo punto è tutto pronto per la fase sperimentale.

Inserite il programma PROVA LM/2 dei listati 5 (Spectrum) e 6 (ZX81).

Rispetto al precedente PROVA LM/1, questo programma contiene due differenze:
— è presente la linea 1 REM 1234567;
— la linea 60 contiene l'istruzione USR 16514 anziché USR 17200.
Analoghe variazioni si verificano per lo Spectrum.
Ora dobbiamo immagazzinare la routine LM negli indirizzi prescelti. Armiamoci di molta pazienza e scriviamo le istruzioni della Tabella 1. Terminato il caricamento è possibile fare una verifica scrivendo, nel caso dello Spectrum:

PRINT PEEK b	ENTER. deve apparire: 42
PRINT PEEK b+1	ENTER. deve apparire: 132

e così via.

Verifichiamo se il nostro programma gira, dando il RUN. Potete notare che il RUN non cancella la routine LM.

Volevamo che la routine fosse anche in qualche modo listabile, per non dover ricorrere a noiosissime PEEK. Ebbene, la routine è listata nella linea 1, che appare stravolta rispetto a come l'avevamo scritta. Si tratta però, come vedremo più avanti, di un LIST molto parziale. Infatti solo con ZX81 è completo mentre con lo Spectrum presenta diverse anomalie.

Esaminiamo la linea 1 che appare con lo ZX81. Con l'aiuto del manuale è possibile ricavare i codici contenuti in questa linea, che sono riportati anche nella tabella 2.

Il primo carattere della linea 1 è E. Il codice di E è 42, ed infatti avevamo scritto nella locazione 16514 il numero 42.

È logico che ora listando la linea 1 il computer scriva il carattere equivalente a 42, cioè E.

La colonna più a destra della tabella 2 (equivalente esadecimale) contiene il numero 42 scritto in esadecimale. (alcune nozioni sui numeri esadecimali sono state trattate nel numero 12).

I vantaggi che si ottengono usando la numerazione esadecimale, sono diversi. Si noterà per esempio che i numeri decimali della seconda colonna sono composti da una, due o anche tre cifre; per esempio 6, 42, 201. Invece tutti i numeri esadecimali della terza colonna sono composti da due caratteri.

```
TABELLA 1

SPECTRUM
LET b= PEEK 23635+256*PEEK
23636 + 5

POKE b,42
POKE b+1,132
POKE b+2,92
POKE b+3,1
POKE b+4,0
POKE b+5,24
POKE b+6,9
POKE b+7,78
POKE b+8,6
POKE b+9,0
POKE b+10,201

ZX81
POKE 16514,42
POKE 16515,14
POKE 16516,64
POKE 16517,78
POKE 16518,6
POKE 16519,0
POKE 16520,201
```

Tabella 1. *Procedura di caricamento della routine LM relativa ai listati 1 e 2.*

```
TABELLA 2

Carattere
della                            Equivalente
linea 1           Codice         esadecimale
E                   42           2A
: (due punti)       14           0E
RND                 64           40
?                   78           4E
▞                    6           06
  (spazio)           0           00
TAN                201           C9
```

Tabella 2. *Routine LM usata con i listati 1 e 2. La colonna a sinistra contiene i caratteri che compaiono nella linea 1 dopo il LIST.*

La terza colonna rappresenta una routine in linguaggio macchina vero e proprio. Un programma LM scritto in numeri esadecimali si chiama CODICE OGGETTO. Lo stesso programma può anche essere scritto sotto forma di istruzioni mnemoniche, come è illustrato nella tabella 3.

Codice oggetto o EXCODES o programma LM	Codice mnemonico o OPCODES o programma Assembly	Equivalente decimale	Equivalente BASIC
2A 0E 40	LD HL, (400E)	42 14 64	LET HL = PEEK 16398 + 256 ★ PEEK 16399
4E	LD C, (HL)	78	LET C = PEEK HL
06 00	LD B, 0	6 0	LET B = 0
C9	RET	201	RETURN

Tabella 3. *Vengono illustrate in questa tabella le istruzioni Assembly equivalenti alla routine LM usata nei programmi 1 e 2, e l'istruzione BASIC equivalenti.*

Linguaggio macchina per Sinclair

Le equivalenze BASIC di questa tabella sono semplicemente esplicative e non producono individualmente il medesimo effetto.

L'istruzione LD HL (400 E) viene codificata in tre byte esadecimali: 2A 4E 00, invece l'istruzione LD C (HL) richiede un solo byte: 4E. Per il momento questo non importa, anche il BASIC richiede 7 byte per immagazzinare il numero 1, ed un solo byte per immagazzinare l'istruzione PRINT, eppure non ci accorgiamo neppure di ciò.

Caricamento automatico di una routine LM

Ora facciamo una prova; verifichiamo se la sequenza di codici esadecimali dello Spectrum:
2A, 84, 5C, 01, 00, 18, 09, 4E, 06, 00, C9
equivale veramente a:
42, 132, 92, 1, 0, 24, 9, 78, 6, 0, 201
e analoga verifica faremo per lo ZX81.

Ovviamente non possiamo caricare i numeri esadecimali con istruzioni del tipo:

POKE 16514, 2a

perché la sintassi dell'istruzione POKE richiede che essa sia seguita da un numero decimale.

Allora dobbiamo utilizzare un programma che trasformi la sequenza di numeri esadecimali nei corrispondenti decimali, e li sistemi in memoria a partire dalla locazione prescelta in poi. Chiameremo questo programma (listati 7 e 8) LOADER/1.

Abbiamo inserito ancora una linea 1 REM per i motivi spiegati in precedenza. Il programma LOADER/1 vero e proprio va dalla linea 10 alla 70.

Solo ZX81: questo programma prende due alla volta i caratteri della stringa H$, ne calcola l'equivalente decimale e lo scrive negli indirizzi del 16514 al 16520. Quando anche gli ultimi due caratteri di H$ sono stati trattati, il programma termina (linea 40).

```
   1 REM 12345678901
  10 FOR K=1 TO 5
  20 LET A= R(RND*8): PRINT AT K,
K; INK A;"█"
  30 NEXT K
  40 PAUSE 100
  50 FOR K=1 TO 5
  60 PRINT AT K,K;USR (PEEK 2363
5+255*PEEK 23636+5)
  70 NEXT K
  80 REM

      PROVA LM/2 - SPECTRUM
```

```
   1 REM 1234567
  10 FOR K=1 TO 5
  20 PRINT AT K,K;CHR$ (INT (RND
*25)+33)
  30 NEXT K
  40 PAUSE 100
  50 FOR K=1 TO 5
  60 PRINT AT K,K; USR 16514
  70 NEXT K
  80 REM

      PROVA LM/2 - ZX81
```

Listati 5 e 6. *La routine LM può venire immagazzinata nella prima linea di programma. In tal caso è possibile registrarla su nastro.*

```
   1 REM *■\???N
?<>
  10 LET IND=PEEK 23635+256*PEEK
 23636+5
  20 DATA "2a","84","5c","01","0
0","18","09","4e","06","00","c9"
,"s"
  30 READ Z$
  35 IF Z$="S" OR Z$="s" THEN ST
OP
  40 LET UNO=CODE Z$-48-(39 AND
Z$>"£")
  45 LET DUE=CODE Z$(2)-48-(39 A
ND Z$(2)>"£")
  50 POKE IND,UNO*16+DUE
  60 LET IND=IND+1
  70 GO TO 30
  80 REM

   PROGRAMMA LOADER/1 - SPECTRUM
```

```
   1 REM 1234567
  10 LET IND=16514
  20 LET H$="2A0E404E0600C9"
  30 POKE IND,16*CODE H$+CODE H$
(2)-475
  40 IF LEN H$=2 THEN STOP
  50 LET IND=IND+1
  60 LET H$=H$(3 TO )
  70 GOTO 30
9999 REM

    LOADER/1          ZX81
```

Listati 7 e 8. *Questi programmi consentono di caricare automaticamente in memoria una routine LM. Nel caso dello ZX81 la routine viene conservata nella variabile stringa H$. Nel caso dello Spectrum in una linea DATA. Occorre notare che i listati relativi allo Spectrum sono scritti in caratteri maiuscoli (basta premere CAPS SHIFT e 2 all'accensione) per ottenere un listato più leggibile. Invece i codici esadecimali della linea DATA e tutti quelli che useremo nel seguito per lo Spectrum, devono essere scritti in caratteri minuscoli. Se fossero maiuscoli la routine di caricamento dovrebbe essere modificata.*

Solo Spectrum: il programma legge i dati contenuti nella linea DATA, e li sistema negli indirizzi da IND (linea 10) in poi. Il programma termina quando il dato letto è uguale a "s".

Dopo il RUN è possibile verificare che:
a) dando il LIST la linea 1 appare così modificata nello Spectrum:

1 REM ★ \ ???Nsedici spazi? < >

Nota: questa linea è di difficile composizione tipografica e può essere ottenuta con ZX Printer. È registrata con il nome "PASSAGGIO 1";
b) con una serie di PRINT PEEK, dal valore iniziale di IND in poi, otteniamo la ben nota serie di numeri:

Spectrum: 42, 132, 92, 1, 0, 24, 9, 78, 6, 0, 201

oppure:

ZX81: 42, 14, 64, 78, 6, 0, 201

Ora possiamo essere convinti di due cose:
a) il codice oggetto contenuto nella stringa H$ o nella linea DATA equivale esattamente alla routine LM scritta in precedenza con i POKE seguiti da numeri decimali;
b) il programma LOADER/1 ci risparmia la scrittura di diverse noiose POKE.

Il programma LOADER/1 ha svolto il suo compito, che consisteva nel caricare la routine LM negli indirizzi prescelti, non ci serve più, e decidiamo quindi di cancellarlo per recuperare spazio in memoria. Attenzione, *non bisogna cancellare la linea 1*, poiché con essa si cancellerebbe anche la routine LM ivi contenuta.

Quindi per cancellare LOADER/1 non bisogna usare il NEW, ma ogni linea va cancellata così:

10 ENTER (o NEWLINE)
20 ENTER (o NEWLINE) ... e così via.

Quando sullo schermo rimane solo la linea 1 è disponibile introdurre il programma utilizzatore della routine LM.

Per provare battete il programma PROVA LM/2 (listati 5 e 6) dalla linea 10 alla linea 70. La linea 1 va ovviamente omessa perché dobbiamo conservare quella ottenuta con LOADER/1.

Dando il RUN la routine LM non viene cancellata. Usando il SAVE la routine viene registrata insieme al programma.

I programmi LOADER/2 e LOADER/3

Adesso caricate il programma LOADER/2 dei listati 10 e 11, date il RUN e per lo Spectrum battete la routine LM della tabella 4, per lo ZX81 quella della tabella 5. Terminato il caricamento premete F, poi: GOTO 100.

Sullo schermo appare: 25.

Solo Spectrum: questa routine:

— scrive il numero 10 nell'indirizzo IND-3;

— scrive il numero 15 nell'indirizzo IND-2;

— somma i due numeri;

— mette il risultato nell'indirizzo IND-1.

Perché IND-3? Perché nel listato di LOADER/2, alla linea 10, B è uguale a:

PEEK 23635 + 256 ★ PEEK 23636 + 8

Notate il + 8, e notate che alla linea 15, IND diventa uguale a B.

Invece la routine LM fissa l'indirizzo di partenza, nel quale scrivere i valori di A e B, all'indirizzo:
PEEK 23635 + 256 ★ PEEK 23636 (istruzione IND) al quale aggiunge solo 5 (istruzioni IND + 6).

Solo ZX81: la routine scrive 10 nell'indirizzo 16514 e 15 nell'indirizzo 16515. Il risultato viene messo nell'indirizzo 16516. I byte 16514, 16515 e 16516 contengono dei dati e non fanno parte della routine di calcolo vera e propria. Questa parte dall'indirizzo 16517, quindi l'istruzione della linea 100 è:

PRINT USR 16517.

Questa istruzione scrive sullo schermo il valore di C, che equivale alla somma dei due numeri cioè 25. Il numero è stato anche scritto nella locazione 16516; con l'istruzione:

ZX81
PRINT PEEK 16516

Spectrum
PRINT PEEK (b−1)

è possibile vedere apparire sullo schermo il secondo numero 25.
Se ora date il LIST, noterete che qualcosa non va nella linea 1.

Per quanto detto in precedenza, essa dovrebbe essere composta così:

```
   1 REM ▀?  5▄RND??7▖7??▗TAN 6789
0123455789012
```

per un totale di 15 byte della linea REM occupati. (I codici da 67 a 127, ed alcuni altri, restituiscono in genere un punto interrogativo). Invece appare così:

```
   1 REM ▀?;5▄RND TAN 67690123455
5789012
```

Solo nove dei quindici byte sono listati nella linea 1.

Questo accade sempre quando nella routine LM sono presenti istruzioni quali, per esempio, 7E. Peraltro, il fatto stesso che tutti i codici da 67 a 127 vengano rappresentati con un punto interrogativo rappresenta già un grave handicap. Per quanto riguarda lo Spectrum, poi, avevamo visto fin dall'inizio che la linea 1 rappresenta un LIST parziale e imperfetto.

Non esistono rimedi a ciò. L'unica cosa che possiamo fare, è aggiungere al programma LOADER/2 alcune linee, che consentano di listare per controllo la routine LM introdotta.

La variazione per lo ZX81 è pubblicata nel numero 12 (Programma LDC, listato 2). Per lo Spectrum la versione è quella del listato 11 (LOADER/3).

```
   1 REM 12345678901234567890123
456789012
  10 LET B=PEEK 23635+256*PEEK 2
3636+8
  15 LET IND=B
  20 LET H$=""
  30 IF H$="" THEN PRINT AT 0,0;
"BATTI IL CODICE OGGETTO"
  40 IF H$="" THEN INPUT H$
  50 IF H$="f" THEN STOP
  52 LET UNO=CODE H$-48-(39 AND
H$>"£")
  55 LET DUE=CODE H$(2)-48-(39 A
ND H$(2)>"£")
  60 POKE IND,UNO*16+DUE
  70 LET IND=IND+1
  80 LET H$=H$(3 TO )
  90 GO TO 30
 100 PRINT USR B
9999 REM
      LOADER/2        SPECTRUM
```

```
   1 REM 12345678901234567890123
456789012
  10 LET IND=16514
  20 LET H$=""
  30 IF H$="" THEN PRINT AT 0,0;
"BATTI IL CODICE"
  40 IF H$="" THEN INPUT H$
  50 IF H$="F" THEN STOP
  60 POKE IND,16*CODE H$+CODE H$
(2)-475
  70 LET IND=IND+1
  80 LET H$=H$(3 TO )
  90 GOTO 30
 100 PRINT USR 16517
9999 REM
      LOADER/2        ZX81
```

Listati 9 e 10. *Questi programmi caricano qualsiasi routine LM composta da numeri esadecimali, che può essere battuta direttamente sulla tastiera.*

Dopo averlo introdotto aggiungete le linee:

Spectrum
210 PRINT USR b
220 PRINT PEEK (b − 1)

ZX81
210 PRINT USR 16517
220 PRINT PEEK 16516

Queste due linee rappresentano il programma BASIC. La linea 210 scrive il risultato della elaborazione effettuata dalla routine LM, cioè la somma dei due numeri contenuti negli indirizzi b − 3 e b − 2 (oppure 16514 e 16515). Tale somma viene anche sistemata nell'indirizzo b − 1 (16516) e con la linea 220 verifichiamo che ciò sia realmente avvenuto.

Date il RUN e battete la routine LM SOMMA 1 (tabelle 4 e 5) un byte per volta, o in gruppi di più byte come preferite. Al termine premete f e sullo schermo appare la lista

Indirizzo	Codice esadecimale	Assembly	Equivalente decimale	Analogia BASIC
IND	2a	LD HL, (5c 53)	42	*LET HL = PEEK 23635 + 256 ★ PEEK 23636*
IND = PEEK 23635 + 256 ★ PEEK 23636 + 5	53		83	
	5c		92	
IND + 3	01	LD BC, 5	1	*LET BC = 5*
	05		5	
	00		0	
IND + 6	09	ADD HL, BC	9	*LET HL = HL + BC*
	e5	PUSH HL	229	*LET PUSH = HL*
IND + 8	3e	LD A, 10	62	*LET A = 10*
	0a		10	
	77	LD (HL), A	119	*POKE HL, A*
	23	INC HL	35	*LET HL = HL + 1*
IND + 12	3e	LD A, 15	62	*LET A = 15*
	0f		15	
IND + 14	77	LD (HL), A	119	*POKE HL, A*
	e1	POP HL	225	*LET HL = PUSH*
	7e	LD A, (HL)	126	*LET A = PEEK HL*
	23	INC HL	35	*LET HL = HL + 1*
	86	ADD A, (HL)	134	*LET A = A + PEEK HL*
	23	INC HL	35	*LET HL = HL + 1*
	77	LD (HL), A	119	*POKE HL, A*
IND + 21	he	LD C, (HL)	78	*LET C = PEEK HL*
	06	LD B, 0	6	*LET B = 0*
	00		0	
IND + 24	c9	RET	201	*RETURN*

Tabella 4 e 5. *Esempio di routine LM che somma due numeri.*

Indirizzo	Codice esadecimale	Assembly	Equivalente decimale	Analogia BASIC
16514	0A		10	*POKE 16514, 10*
16515	0F		15	*POKE 16515, 15*
16516	00		0	*POKE 16516, 0*
16517	21	LD HL, 4082	33	*LET HL = 16514*
	82		130	
	40		64	
16520	7E	LD A, (HL)	126	*LET A = PEEK HL*
16521	23	INC HL	35	*LET HL = HL + 1*
16522	86	ADD A, (HL)	134	*LET A = A + PEEK HL*
16523	23	INC HL	35	*LET HL = HL + 1*
16524	77	LD (HL), A	119	*POKE HL, A*
16525	4E	LD C, (HL)	78	*LET C = PEEK HL*
16526	06	LD B, 0	6	*LET B = 0*
	00		0	
16528	C9	RET	201	RETURN

di controllo della routine LM: quella che può essere vista nella tabella 6. Premendo CONT si ottengono successive visualizzazioni. Con un ultimo CONT si mandano in esecuzione le linee 210 e 220 e sullo schermo compaiono i due numeri 25.

Una procedura diversa per lo Spectrum

Gli utilizzatori dello Spectrum avranno notato quanto sia noiosa la procedura seguita per immagazzinare una routine LM nella prima linea del programma BASIC, soprattutto perché l'area del program-

```
        TABELLA 6    (Spectrum)
 Lista di controllo della routine
              SOMMA/1

23760      10       0a       ?
23761      15       0f       ?
23762       0       00       ?
23763      42       2a       *

23764      83       53       S
23765      92       5c       /
23766       1       01       ?
23767       5       05       ?
23768       0       00       ?
23769       9       09
23770     229       e5  RESTORE
23771      62       3e       >
23772      10       0a       ?
23773     119       77       w
23774      35       23       #
23775      62       3e       >
23776      15       0f       ?
23777     119       77       w

23778     225       e1    LLIST
23779     126       7e       "
23780      35       23       #
23781     134       86       ▞
23782      35       23       #
23783     119       77       w
23784      78       4e       n
23785       6       06
23786       0       00       ?
23787     201       c9      <>
```

Tabella 6. *Videata di controllo offerta dal programma LOADER/3 quando viene caricata la routine SOMMA 1.*

ma non ha un indirizzo fisso di partenza.

Dobbiamo trovare un metodo che consenta di scrivere la routine partendo da un indirizzo fisso. Questo obiettivo può essere raggiunto solo tornando a scrivere la routine nella parte alta della memoria RAM, e con lo Spectrum possiamo farlo; la routine LM può essere registrata su nastro anche se risiede nella parte alta.

Occorre però usare una procedura che a prima vista può apparire abbastanza complessa, ma che diventa semplice con un po' di pratica.

Anzitutto dobbiamo scegliere un adeguato indirizzo di partenza nella parte alta della memoria RAM.

Il manuale suggerisce l'indirizzo 32500, ed effettivamente è il migliore.

Scegliendo l'indirizzo 32500 come locazione di partenza per la routine LM abbiamo a disposizione 100 indirizzi: tutti quelli fino al 32599.

Infatti il 32599 è il byte RAMTOP, cioè il byte più alto della parte di RAM a disposizione del BASIC, quindi l'ultimo byte utilizzabile della RAM.

ALCUNE NOZIONI SUI NUMERI BINARI

Come si legge un numero binario

La lettura di un numero binario inizia dall'ultima cifra a destra, che può valere 1 oppure 0. Nel numero binario:

10011101

l'ultima cifra a destra è 1 quindi vale 1. Invece nel numero:

11110010

l'ultima cifra a destra è zero quindi vale 0.
Tutte le altre cifre procedendo sempre da destra verso sinistra valgono sempre zero, in qualsiasi posizione si trovino, se sono zero. Se invece sono 1 il loro valore si calcola con successive potenze del 2. Nel numero binario:

10011

le cifre valgono:

16 0 0 2 1

quindi il numero nel suo complesso vale:

$$16 + 0 + 0 + 2 + 1 = 19$$

Il numero binario:

11111111

vale:

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

Somma di due numeri binari

Vediamo ora come avviene la somma di due numeri binari. Anche loro vanno messi in colonna uno sotto l'altro. Per esempio:

```
 0011 +
 0101 =
```

L'addizione nell'esempio rappresenta tutte le combinazioni possibili tra le cifre 1 e 0. Volendo sommare singolarmente ogni colonna partendo dalla prima a destra senza tener conto dei riporti, avremmo i risultati illustrati nella tabella 8.
Nella effettiva esecuzione della addizione dobbiamo tenere conto dei riporti, ma la tabella 8 rimane valida. L'addizione risolta appare così:

```
 111
 0011 +
 0101 =
 1000
```

la prima colonna a destra dà 0; il carry vale 1 e lo annotiamo sulla colonna subito a sinistra.
Questa seconda colonna (1 − 1 − 0) può essere considerata equivalente a 1 − 1 ignorando lo zero. Quindi anche la seconda colonna dà 0 e il carry vale ancora 1.
Analogamente nella terza colonna (0 − 1 − 1) consideriamo solo i due 1 ignorando lo 0 ed il risultato è ancora 0 con un carry uguale a 1.
La quarta colonna (1 − 0 − 0) può essere considerata uguale a 1 − 0 o più semplicemente 1, e dà 1 senza alcun carry.
La somma dei due numeri è 1000. In decimale:

```
0011 = 3
0101 = 5
1000 = 8
```

Vediamo ora un altro caso:

```
 111 +
 101 =
```

La prima colonna a destra dà 0 con il carry uguale a 1. Lo scriviamo sulla seconda colonna.
Questa dà 0 con il carry a 1 che viene scritto sulla terza. A questo punto la situazione è:

```
 11
 111 +
 101 =
  00
```

Linguaggio macchina per Sinclair

Sopra RAMTOP ci sono ancora 169 byte di memoria RAM, dal 32600 al 32768 che sono destinati a contenere i caratteri definiti dall'utente.

Perché il byte RAMTOP è posto sotto l'area dei caratteri-utente?

La risposta è semplice: perché tutti i byte sopra la RAMTOP sono al sicuro rispetto eventuali incidenti che potrebbero accadere al di sotto. Non risentono il LOAD, né il SAVE, né il NEW; non possono assolutamente confondersi con il programma BASIC perché RAMTOP è il limite della memoria utilizzabile dal BASIC.

Scrivendo un programma BASIC molto lungo, questo può riempire tutta la RAM ma solo fino al byte RAMTOP e neppure un byte di più.

Potete fare una prova abbassando il byte RAMTOP all'inizio della memoria RAM. L'abbassamento si ottiene con l'istruzione: CLEAR seguita dall'indirizzo del byte in cui si vuole posizionare la RAMTOP. Per esempio, CLEAR 25000 significa:

L'utente può utilizzare al massimo i byte fino al 24999.

Rimane inteso che i byte dal 25000 al 32768 esistono ancora e possono essere scritti o letti con istruzioni del tipo PEEK o POKE. Scrivete:

CLEAR 23860

e poi ENTER. Provate poi a scrivere la linea:

1 REM 00000

Arrivati al quarto zero sentirete il segnale acustico di MEMORIA PIENA, e non riuscirete a scrivere il quinto zero. Se premete ENTER appare il segnale: no room for line. Eppure non avete ancora introdotto neppure una linea!

Dunque non è mai conveniente spostare RAMTOP troppo in basso, ci converrà abbassarlo solo di 100 byte rispetto al suo valore normale.

Con CLEAR 32499, si ottengono 100 byte protetti, compresi tra l'indirizzo 32500 e il 32599. In questi

Seguito riquadro

I tre 1 della terza colonna vanno risolti separatamente. I primi due danno:

```
  1 +
  1 =
 ---
 10
```

Aggiungendo il terzo otteniamo:

```
 10 +
  1 =
 ---
 11
```

Quindi il risultato finale diventa:

```
  11
 111 +        7 +
 101 =        5 =
 ----  ----  ---
1100         12
```

```
   1 REM 1234567890123456789012345678901
2
  10 LET B=PEEK 23635+256*PEEK 23636+8
  15 LET IND=B
  20 LET H$=""
  30 IF H$="" THEN PRINT AT 0,0;IND;" ?(f=STOP)"
  40 IF H$="" THEN INPUT H$
  50 IF H$="f" THEN GO TO 100
  52 LET UNO=CODE H$-48-(39 AND H$>"£")
  55 LET DUE=CODE H$(2)-48-(39 AND H$(2)>"£")
  60 POKE IND,UNO*16+DUE
  70 LET IND=IND+1
  80 LET H$=H$(3 TO )
  90 GO TO 30
 100 FOR K=B TO IND-1
 110 LET H$=""
 120 LET N=PEEK K
 130 FOR W=1 TO 2
 135 LET P=(48+N-16*INT (N/16))
 140 LET H$=CHR$ (P+(39 AND P>57))+H$
 150 LET N=INT (N/16)
 160 NEXT W
 180 PRINT K;TAB 7;PEEK K;TAB 12;H$;TAB 16;CHR$ PEEK K
 190 NEXT K
 200 STOP
9999 REM
     LOADER/3     SPECTRUM
```

Listato 11. *Oltre a caricare qualsiasi listato in linguaggio macchina (massimo 32 byte) questo programma offre anche, al temine del caricamento, un listato di controllo simile a quello illustrato nella tabella 6.*

```
   1 CLEAR 32499
  10 LET IND=32500
  20 LET H$=""
  30 IF H$="" THEN PRINT AT 0,0;IND;" ?(f=STOP)"
  40 IF H$="" THEN INPUT H$
  50 IF H$="f" THEN GO TO 100
  52 LET UNO=CODE H$-48-(39 AND H$>"£")
  55 LET DUE=CODE H$(2)-48-(39 AND H$(2)>"£")
  60 POKE IND,UNO*16+DUE
  70 LET IND=IND+1
  80 LET H$=H$(3 TO )
  90 GO TO 30
 100 FOR K=32500 TO IND-1
 110 LET H$=""
 120 LET N=PEEK K
 130 FOR W=1 TO 2
 135 LET P=(48+N-16*INT (N/16))
 140 LET H$=CHR$ (P+(39 AND P>57))+H$
 150 LET N=INT (N/16)
 160 NEXT W
 180 PRINT K;TAB 7;PEEK K;TAB 12;H$;TAB 16;CHR$ PEEK K
 190 NEXT K
 210 PRINT USR 32503
 220 PRINT PEEK 32502
9999 REM
 LOAD AND CHECK - SPECTRUM
```

Listato 12. *Questo programma consente di caricare nello Spectrum routine LM, sistemandole negli indirizzi compresi tra il 32500 e il 32599. Con la procedura descritta nell'articolo la routine (massimo 100 byte, facilmente aumentabili) pur essendo sistemata nella parte alta della memoria può essere registrata su nastro.*
Si evita così di usare la linea 1, dato che nello Spectrum l'area del programma non ha un indirizzo fisso di partenza.

byte possiamo sistemare routine LM anche molto lunghe.

Il programma LOAD AND CHECK (abbreviato: LDC) del listato 12 tiene conto di quanto detto finora: le routine LM vengono immagazzinate a partire dal byte 32500.

Ora vediamo come è possibile scrivere la routine LM negli indirizzi dal 32500 in avanti, riuscendo poi a registrarla su nastro.

Per cominciare caricate il programma LDC del listato 12 e aggiungete le linee:

210 PRINT USR 32503
220 PRINT PEEK 32502

Date il RUN e battete il codice esadecimale della routine LM illustrata nella tabella 7 (SOMMA 2). Le prime tre istruzioni:

0 A, 0F, 00

scrivono i numeri 10, 15 e 0 nelle locazioni 32500, 32501 e 32502.

La routine somma il contenuto delle locazioni 32500 e 32501, quindi

Indirizzo	Codice esadecimale	Assembly	Equivalente decimale	Analogia BASIC
32500	0a		10	*POKE 32500, 10*
	0f		15	*POKE 32501, 15*
	00		0	*POKE 32502, 0*
32503	21	LD HL, 7EF4	33	*LET HL = 32500*
	f4		244	
	7e		126	
32506	7e	LD A, (HL)	126	*LET A = PEEK HL*
32507	23	INC HL	35	*LET HL = HL + 1*
32508	86	ADD A, (HL)	134	*LET A = A + PEEK HL*
32509	23	INC HL	35	*LET HL = HL + 1*
32510	77	LD (HL), A	119	*POKE HL, A*
32511	4e	LD C, (HL)	78	*LET C = PEEK HL*
32512	06	LD B, 0	6	*LET B = 0*
	00		0	
32514	C9	RET	201	*RETURN*

Tabella 7. *Esempio di routine LM che può essere caricata con il programma LDC del listato 12.*

l'istruzione USR ritorna il risultato, cioè il numero 25.

La linea 210 fa partire la routine dall'indirizzo 32503, perché gli indirizzi precedenti contengono i dati da sommare.

Il risultato viene anche scritto nella locazione 32502, e con la linea 220 si verifica che ciò sia realmente accaduto.

Terminato il caricamento della routine premiamo f ed appare il LIST, simile a quello già visto nella Tabella 6 per SOMMA 1.

Premendo due volte CONT il computer esegue il programma (cioè le linee 210 e 220) scrivendo sullo schermo:

25
25

Ammettiamo ora di voler registrare questo programma, per poterlo utilizzare in un secondo tempo. Anzitutto potremmo cancellare tutte le linee dalla 10 alla 200 (eliminando così il programma LDC che non serve più e lasciando solo il programma BASIC che utilizza la routine, cioè le linee 210 e 220), ma siccome abbiamo sufficiente memoria, lasciamo LDC al suo posto.

La procedura da seguire per la registrazione è questa:

a) cambiare così la linea 1 del programma già pronto in memoria:

0 CLEAR 32499: LOAD “ ” CODE 32500, 100: GOTO 210

Attribuiamo alla linea il numero zero per renderla sicura da eventuali cancellazioni effettuate per errore. Vedremo tra poco come ottenere la linea 0, per ora scrivetela con il numero 1.

Il programma e la routine vanno registrati separatamente, ma l'istruzione della linea 0 oltre ad abbassare il byte RAMTOP al valore 32499, consentirà di caricare il programma e la routine LM insieme. In pratica programma e routine saranno registrati sul nastro uno di seguito all'altro, con due operazioni di SAVE, ma in fase di caricamento, sarà sufficiente una sola operazione di LOAD.

b) Eseguire la registrazione del programma BASIC, scrivendo:

SAVE “somma” LINE 0

Avviate il registratore, premete ENTER e un tasto qualsiasi. Il programma BASIC viene registrato. Al termine, fermate il registratore possibilmente con il tasto PAUSE.

c) Eseguire la registrazione della routine LM, scrivendo:

SAVE “somma” CODE 32500, 100

Avviate nuovamente il registratore ed eseguite la seconda registrazione. Ora programma e routine sono pronti sul nastro. Spegnete il computer e riaccendetelo, per avere la certezza che la memoria sia completamente vuota.

Riavvolgete il nastro e scrivete:
LOAD “somma” o anche: LOAD “ ”

Osservando lo schermo noterete che il computer effettua il caricamento del programma. Terminato questo non si ferma ma comincia automaticamente il caricamento della routine LM che segue sul nastro. Ecco la spiegazione:

Con l'istruzione SAVE “somma” LINE 0 registriamo il programma con un particolare accorgimento: facciamo in modo che il programma, appena caricato, parta automaticamente dalla linea 0, di cui parleremo tra poco.

Cioè SAVE seguito da LINE 0 fa in modo che ogni LOAD eseguito successivamente diventi uguale a LOAD seguito automaticamente da GOTO 0.

Una volta caricato, il programma va automaticamente in esecuzione partendo dalla linea 0. Questa abbassa il byte RAMTOP per creare un'area protetta ed esegue un LOAD particolare, cioè carica dei dati binari (contenuti nella seconda registrazione) e li sistema a partire dal byte 32500 in avanti.

Terminato questo secondo caricamento il programma passa alla linea 210 per evitare che la linea 10 chieda l'inserimento di una nuova routine LM.

Se il programma LDC era stato cancellato in precedenza, il GOTO 210 può essere omesso ed il programma BASIC prosegue dalla prima linea successiva allo 0.

La linea 0 presenta il grosso vantaggio di poter essere listata, ma di non poter essere cancellata per errore, né editata.

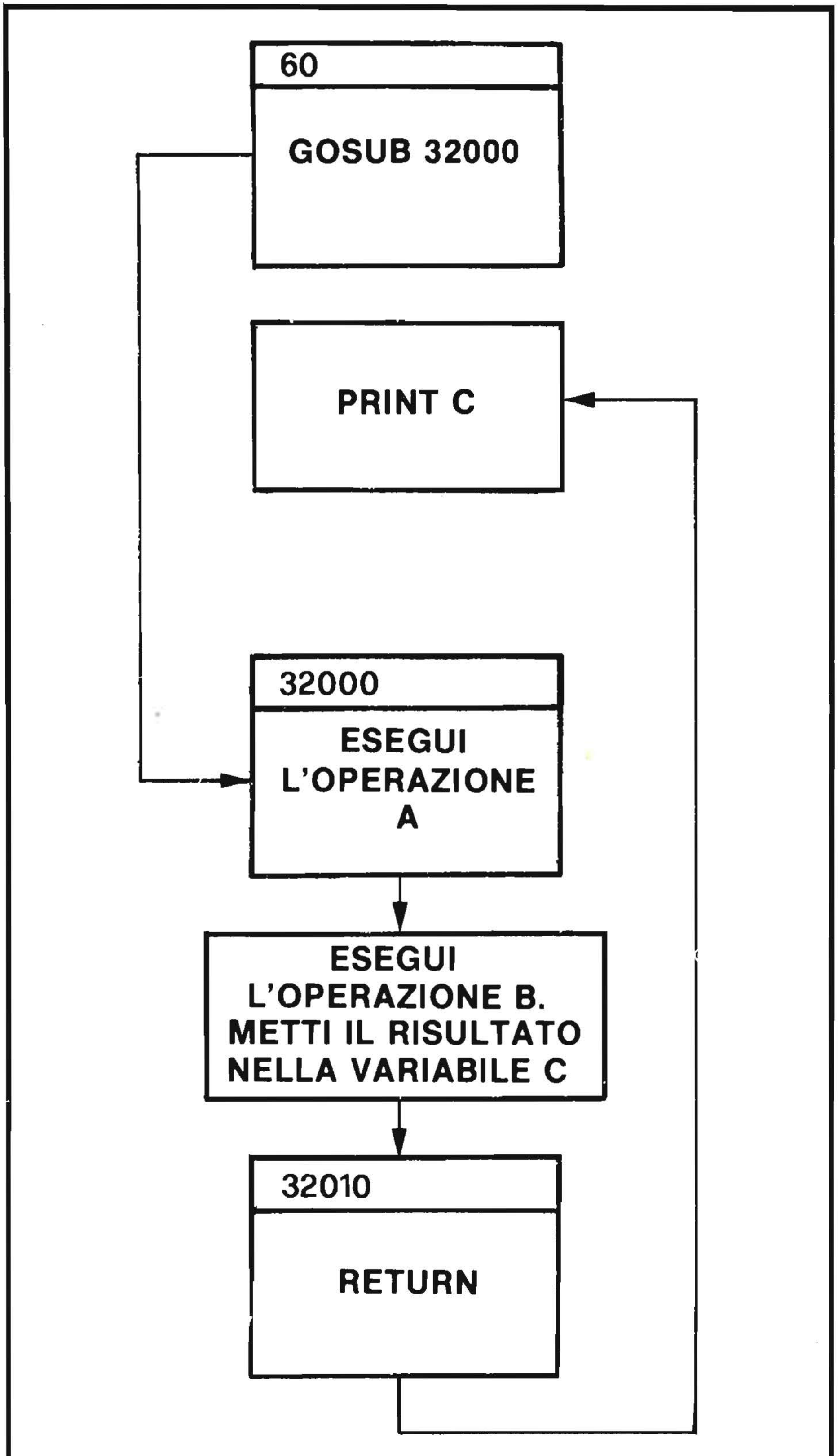


Figura 3. *Rappresentazione grafica della istruzione PRINT USR 32000 contenuta nella linea 60 del listato 1.*

Numeri			Risultato	Carry
1	1		0	1
0	1		1	0
1	0		1	0
0	0		0	0
1	0	0	1	0
0	0	0	0	0
1	1	1	11	0

Tabella 8. *Combinazioni possibili tra i digit 1 e 0 quando viene eseguita una addizione binaria.*

Una volta inserita è come se non ci fosse perché il programma può cominciare normalmente dalla linea 1; tuttavia la linea 0 c'è e assicura che venga caricata sempre anche la routine LM.

È importante ricordarsi di usare SAVE "prog" LINE 0 quando si registra il programma BASIC. La linea 0 può anche essere usata in altre occasioni per proteggere le routine LM contenute nella linea 1 REM (che diventa 0 REM) o per contenere messaggi particolari, come per esempio il nome dell'autore di un programma.

Eccome come ottenere il numero di linea 0:

1) scrivete una linea 1. Per esempio:

```
1 REM CLIVE SINCLAIR
```

2) Battete il comando diretto:

```
POKE (PEEK 23635 + 256 ★
     PEEK 23636 + 1), 0
```

e premete ENTER. Lo schermo si sbianca: se date il LIST vedete che la linea 1 è diventata linea 0, e volendo potete inserire una nuova linea 1. Se in seguito decidete di modificare la linea 0, dovete ritrasformarla in linea 1. Ciò si ottiene con:

```
POKE (PEEK 23635 + 256 ★
     PEEK 23636 + 1), 1
```

Se avevate inserito una linea 1, avrete sullo schermo due linee 1.
La linea 0 può essere ottenuta anche sullo ZX81. L'istruzione è:

```
POKE 16510, 0
```

Per ritrasformarla in linea 1 occorre scrivere:

```
POKE 16510, 1
```

Il Carry

Cominciamo ora la spiegazione di alcuni concetti importantissimi che vanno assolutamente compresi se si vuole programmare sul serio in linguaggio macchina. Ora parleremo del Carry, nella prossima puntata tratteremo l'AREA STACK e l'istruzione JUMP.

Ammettiamo di voler sommare i numeri decimali 1883 e 4542.

La somma va eseguita partendo dalle due cifre incolonnate più a destra. Il primo passo nello svolgimento dell'addizione è il seguente:

```
4882 +
4542 =
----
   5
```

Nel secondo passo dobbiamo sommare 8 e 4. Il risultato è 12, ma non possiamo scrivere 12 sotto la seconda colonna perché ogni colonna può contenere una sola cifra. Allora, come abbiamo imparato a scuola, dividiamo il 12 in due parti. Le unità (2) vanno scritte sotto la colonna in oggetto, le decine (1) costituiscono il RIPORTO (in inglese, CARRY) e le annotiamo in caratteri più piccoli sopra la colonna successiva verso sinistra. Ora la situazione è la seguente.

```
  1
1883 +
4542 =
----
  25
```

Sommando la terza colonna dobbiamo tenere conto di tre numeri: 1 (riporto), 8 e 5. La loro somma è 14. Anche qui abbiamo un riporto e scriviamo:

```
 11
1883 +
4542 =
----
 425
```

Il risultato finale è 6425.

Se vogliamo essere pignoli, anche la prima colonna, quella più a destra, aveva generato un riporto uguale a zero ed anche l'ultima termina con un riporto uguale zero. Quindi c'è un riporto per ogni cifra e cioè: 0 1 1 0. Uno (o altro valore) quando c'è un riporto, zero quando il riporto non c'è.

Ora, dobbiamo sapere che nessuno è più pignolo di un computer e mentre noi ignoriamo il riporto quando è uguale a zero, il computer invece lo considera e lo annota anche in quel caso.

Per eseguire qualsiasi somma il computer non segue il procedimento appena visto, perché era relativo ai numeri decimali mentre lui lavora con i numeri binari. Ma anche le addizioni con i numeri binari comportano spesso dei riporti ed il computer deve annotare in qualche parte della sua memoria (come noi lo annotiamo sulla carta) i riporti delle operazioni che esegue.

Sommando due numeri in aritmetica binaria il riporto può valere solo 0 (quando non c'è) oppure 1 (quando c'è).

Siccome zero e uno sono anche i valori possibili di un bit, il computer usa un bit (non un byte) per conservare i dati relativi al riporto. Questo bit è contenuto nel registro F, che non è altro se non un byte della CPU.

Anche altri bit del registro F vengono utilizzati per indicare particolari condizioni.

I bit di F si contano da 0 a 7 e quello usato per indicare lo stato di CARRY (= RIPORTO 1) o NON CARRY (= NIENTE RIPORTO) è il bit numero 0.

Proprio perché questi bit del registro F indicano qualche cosa, si chiamano FLAGS (INDICATORI) e sono tutti contenuti nello stesso registro. D'ora in poi chiameremo il riporto "CARRY", come consuetudine.

Alcune nozioni sui numeri binari vengono illustrate nel riquadro relativo e nella tabella 8. Quando il computer esegue un'addizione sistema il primo numero da sommare nell'accumulatore o Registro A (occorre notare che questo registro è un byte, quindi può contenere al massimo numeri fino a 255. Numeri più alti possono essere sistemati in un registro doppio).

Aggiunge poi al numero nell'accumulatore il secondo numero da sommare ed al termine dell'operazione l'accumulatore contiene la somma. Se questa supera 255 abbiamo un owerflow, e il flag del carry (il bit 0 del registro F) viene messo a 1. Invece l'accumulatore assume un valore pari alla differenza tra la somma e 256.

Per esempio, se proviamo a mettere 255 nell'accumulatore sommandogli poi uno, l'accumulatore va a zero ed il flag del carry viene messo al valore di 1.

Mettendo invece nell'accumulatore 200 e sommandogli 100, il flag del carry viene messo a 1 e l'accumulatore diventa uguale a 300 − 256 = 44.

Sommando due numeri che danno meno di 256 il FLAG C (= CARRY) viene resettato (= 0).

L'istruzione: ADD A, B (tabella 4 nel numero 12) può essere considerata equivalente alle seguenti istruzioni BASIC:

```
LET A = A + B
LET CARRY = INT ((A + B)/256)
```

Si noti che la gestione del carry tra un bit ed il successivo dello stesso byte (durante un'operazione aritmetica) è automatica e non può essere controllata nemmeno dal linguaggio macchina.

Il bit di carry (del registro F) serve per ottenere il riporto corretto sommando numeri di più byte. Chi è molto curioso può provare ad affrontare le pagine relative in un libro di programmazione in LM per la Z80.

■

Abukir 1798 battaglia navale per CBM

— Parte prima —

Pubblichiamo in questa prima puntata una splendida simulazione di una battaglia navale

di *Umberto Giovanni Barzaghi*

Ho già parlato, sulle pagine della rivista Jackson consorella di **Personal Software** - **Bit** - di wargame, boardgame e computer-game, e in particolare, sulle differenze intercorrenti tra l'uno e l'altro di essi. Ho anche realizzato due programmi che esemplificano due dei tre tipi suddetti: un classico computer-game (Caccia all'U-boot su **Bit** n. 10 anno 3°) ed un boardgame piuttosto sofisticato (forse troppo! '14-'18, su **Bit** n. 23 anno 4°).

Mi accingo ora ad offrire al pubblico di **Personal Software** la mia terza creatura nel campo dei "giochi di guerra" e che, almeno nelle mie intenzioni, dovrebbe avere pieno diritto a fregiarsi del titolo di wargame, e ad acquisire i pregi dei due tipi suddetti (l'esperienza acquisita nella realizzazione di entrambi mi è stata assai utile nella progettazione di questo programma) senza accusarne i difetti; non è infatti un semplice boardgame, come '14-'18, poiché non si gioca su di una scacchiera ma su di una cartina geografica ricca di particolari; inoltre Abukir 1798 non presenta quelle caratteristiche di lentezza che rendevano '14-'18 esasperante da giocare per chiunque non fosse naufragato su di un'isola deserta con la sola compagnia della copia di **Bit** su cui il programma è apparso!

Inoltre il programma in questione presenta, come in Caccia all'U-boot, una parte, per così dire, tattica, in cui cioè non si debbono prendere delle decisioni strategiche a medio o lungo termine nell'economia del gioco (come spostare le truppe o - in questo caso - le navi in modo da affrontare poi lo scontro tattico) ma delle decisioni a breve e brevissimo termine (che alzo dare ai cannoni, dove e quando piazzare il colpo).

Si può anzi affermare che Abukir 1798 è frutto diretto del desiderio di realizzare una sintesi di gioco tattico e gioco strategico, oltre che di una serie di particolari circostanze. Una di queste, come spesso accade, è legata ai miei studi di ingegneria, ed in particolare ad una formula di dinamica balistica studiata in Meccanica Razionale e rimasta appesa in qualche angolo della memoria finché non è venuto il momento di utilizzarla (per un gioco!). Un'altra di queste circostanze favorevoli è rappresentata da un libro: "Le grandi battaglie navali a vela" di Christopher Lloyd, edito dalla Rizzoli nella collana Internationl Library, su cui ho trovato cartine e prospetti di cui mi sono servito per realizzare il programma.

Penso che sia venuto il momento di svelare cosa si nasconde dietro il titolo del programma. Abukir è il nome di una baia sulla costa mediterranea dell'Egitto a est di Alessandria, dove, il 1 Agosto del 1798 la flotta inglese comandata dall'ammiraglio Nelson sorprese ed in gran parte affondò o costrinse ad arenarsi la flotta francese di appoggio alla campagna di Napoleone in Egitto, comandata dall'ammiraglio Brueys.

Avrei, naturalmente, potuto scegliere una battaglia più nota, ad esempio Trafalgar o Capo S. Vincenzo o Camperodown; ma ci sono alcune ragioni che mi hanno fatto scegliere la battaglia della baia di Abukir.

Una è senz'altro la varietà e l'interesse del campo di battaglia, chiuso da tre lati su quattro dalla terraferma e dalle secche che ne facevano un approdo ideale per una flotta di appoggio. Un'altra ragione è data dal fatto che alla vittoria in questa battaglia è legata la fama di Horatio Nelson, più che alla battaglia di Trafalgar, in cui l'ammiraglio inglese perse la vita, per una fucilata di un tiratore scelto piazzato sulla coffa di una nave nemica solo mezz'ora dopo l'inizio della battaglia.

Un'ultima ragione è data dal fatto che l'eliminazione della flotta d'appoggio francese, segnò la fine della campagna di Napoleone in Egitto, intrappolando il corpo di spedizione francese e costringendo il piccolo caporale ad abbandonare i suoi uomini fuggendo in Franca su di una veloce fregata.

Un'altro elemento di interesse è dato dalla disparità delle forze in campo, nettamente a favore dei francesi, che si fecero però sorprendere all'ancora dalla manovra aggirante degli inglesi.

La flotta di Nelson era composta da dodici navi gemelle, dodici due ponti, comandate dai suoi migliori capitani, la famosa "banda di fratelli" (così chiamata da un famoso verso dell'Enrico IV di Shakespeare "...We, a bunch of brothers".). In realtà una di queste, la "Culloden", che era rimasta indietro rispetto al grosso, nella fretta di serrar sotto, aveva finito per incagliarsi in un banco di sabbia al largo della penisola di Abukir; solo a battaglia finita sarebbe stata disincagliata.

Nel programma ho preferito porre la "Culloden" in una posizione leggermente arretrata (come risulta,

Abukir 1798 battaglia navale per CBM

guarda caso, dalla cartina della battaglia tratta dall'altro testo di cui mi sono servito: la solita "Storia delle guerre" del Feldmaresciallo - inglese! - Bernard Law Montgomery, il famoso Montgomery di El Alamein), prima che si arenasse, ma in grave pericolo di farlo.

La flotta francese, invece, contava, oltre che su dodici dueponti, su di un treponti, la nave ammiraglia "L'Orient", e quattro fregate, schierate nella parte più interna della baia, protette dal resto della flotta francese. Il nodo dell'intera battaglia è rappresentato prorio dalla posizione della flotta francese all'ancora, rispetto alle secche.

Tutto sarebbe andato bene, se la nave in testa fosse stata ancorata più vicino ai bassifondi di fronte alla punta di Abukir e se le altre navi fossero state più ravvicinate. Le prime sei navi inglesi, infatti, si incunearono tra i bassifondi, in un punto in cui l'acqua era profonda solo cinque braccia, e la prima delle navi francesi, in altre parole penetrando dietro la linea nemica, dalla parte in cui i cannoni francesi non erano stati preparati.

Il programma

Il programma consente di giocare contro il calcolatore. Coloro che hanno avuto già modo di giocare con '14-'18 non si spaventino, contrariamente al programma suddetto Abukir ha una routine di scelta delle mosse del calcolatore assai rapida, per quanto altrettanto efficiente di quella di '14-'18; ciò è stato possibile grazie ad una notevole differenza regolamentare tra i due wargame: mentre in '14-'18 ogni giocatore doveva scegliere, di volta in volta, non solo che mossa effettuare tra mezza dozzina di alternative possibili, ma anche quale pedina muovere; in Abukir 1798, come nella maggior parte degli attuali wargame il giocatore deve muovere alternativamente ciascun pezzo, aumentando così il realismo della simulazione: trattandosi infatti di una battaglia navale, non avrebbe molto senso che una nave, che non si trovasse all'ancora, rimanesse ferma nella sua posizione rinunciando al proprio turno, mentre le altre procedono a normale velocità; per cui, anche nel caso in cui si "passi" la propria mossa, la nave continua ad avanzare nella sua rotta attuale. Poiché la flotta francese è composta da un numero di navi maggiore rispetto a quella inglese, per mantenere una velocità costante tra le due flotte, le due flotte si muovono a mosse alternate a partire dalle navi di testa dello schieramento (la "Goliath" per la flotta inglese ed il primo dei dueponti per la francese), fino alla mossa della "Culloden" e del dueponti francese numero dodici; dopo di che per la flotta inglese toccherebbe nuovamente alla "Goliath", ciò renderebbe però la velocità della flotta inglese lievemente superiore a quella della flotta francese, differenza del tutto ingiustificata. Quindi dopo il dueponti francese numero dodici, la flotta francese può muovere la "Généreux" e le quattro fregate prima di restituire la mossa alla flotta inglese. Per la stessa ragione, quando, nell'arco dello scontro si creano dei vuoti nelle file delle due flotte, questi "gap" non vengono colmati dalle navi seguenti nella linea di fila e può capitare, a causa di rese o affondamenti o disalberamenti, che la mossa tocchi a due o più navi della stessa flotta consecutive.

Poiché il calcolatore deve solo scegliere "come" muovere ciascun pezzo e non "quale" pezzo muovere, la routine che determina la mossa del calcolatore è piuttosto rapida, pur mantenendo un grado di efficacia elevato quanto quello della analoga routine di '14-'18.

Contrariamente a quest'ultimo programma il giocatore può scegliere quale delle due flotte controllare, anche se deve mantenere la sua scelta per la durata di tutta la partita. Analogamente ad altri programmi da me realizzati, è possibile salvare su cassetta una situazione intermedia della partita in modo da poterla riprendere in un momento successivo.

Il programma inizia infatti offrendo all'utente la possibilità di riprendere una partita precedentemente interrotta; nel caso in cui l'utente risponda positivamente, si accede ad una opportuna subroutine di caricamento e la partita viene ripresa nel punto in cui era stata interrotta, dopo aver ricostruito sulla cartina la situazione esistente. Nel caso contrario viene richiesto al giocatore quale delle due flotte desidera controllare; quindi appare sul video il campo di battaglia con la posizione iniziale delle due flotte, così come l'ho ricavata dal libro di Lloyd. La flotta francese, all'ancora, può rispondere al fuoco se coinvolta in uno scontro con un dueponti inglese, ma, per poter salpare le ancore, deve saltare il turno, rimanendo immobile.

Il programma può essere diviso in due parti: una parte strategica che si svolge sulla cartina suddetta, ed una parte tattica.

Strategia

La parte strategica è rappresentata dalle evoluzioni delle navi sul

Abukir 1798 battaglia navale per CBM

campo di battaglia ed ha come fine quello di giungere in posizione favorevole alla parte tattica, vale a dire agli scontri balistici, che sono i diretti responsabili dell'esito della battaglia.

Come detto, questa parte si svolge sulla cartina stilizzata rappresentante la baia di Abukir. In grigio (carattere retinato) viene rappresentata la terraferma, mentre in bianco sono indicate le secche. Tutte le navi, naturalmente, si incagliano sulla terraferma; mentre le sole fregate (poiché pescano poco) sono in grado di avanzare sui banchi di sabbia senza arenarsi.

La flotta francese è rappresentata dai pallini scuri, mentre quella inglese dai pallini bianchi. Sul campo di gioco troviamo anche altri due caratteri analoghi a quelli che contraddistinguono le navi, ma piazzati in terraferma: il pallino bianco in negativo piazzato vicino al lato ovest della cartina, rappresenta il forte di Abukir, che non intervenne mai, comunque, nella battaglia - come il lettore avrà modo di constatare di persona giocando, è praticamente impossibile portare un vascello a tiro del forte senza incagliarsi -; interviene invece, nel solo caso in cui uno dei dueponti inglesi giunga a tiro, il forte piazzato sull'isola di Abukir e rappresentato dal pallino nero in negativo nei pressi del lato nord della cartina. Il suo campo di tiro è rappresentato da un quadrato di cinque unità video per lato, centrato sul forte stesso.

Sarà il calcolatore stesso a preoccuparsi di saltare alla pagina tattica del gioco nel caso in cui un giocatore sbadato, giocando ovviamente con la flotta britannica, si andasse a ficcare a tiro delle batterie del forte (il calcolatore, infatti, non commette da questo punto di vista nessun errore e, reiterati tentativi dell'autore di costringere una o più navi avversarie a portarsi a tiro del forte sono miseramente falliti).

Dopo aver dato il via alla partita od averne ripresa una interrotta, il calcolatore provvede a ricordare al giocatore quali sono le mosse consentite e a quali tasti sono associate. Il giocatore può, innanzi tutto, "passare" la propria mossa (Q). In questo caso, però, la nave procede di una unità video nella direzione della sua flotta attuale ed è quindi soggetta alle normali evenienze del gioco; può cioè arenarsi o incagliarsi (solo per le fregate francesi) o entrare nel tiro delle artiglierie del forte (solo per le navi inglesi) o entrare in contatto balistico con una nave avversaria. Quest'ultimo caso si presenta, quando lo spostamento porta la nave ad avere una nave avversaria che non si sia precedentemente arresa in una Z.O.C. (Zone of Control, zona di controllo) pari a un quadrato di lato tre unità video, centrato nella nave in questione, a patto che le rotte delle due navi non siano perpendicolari; ciò per ragioni di congruenza storica, poiché le navi a vela non portavano che pochi pezzi "in caccia", vale a dire a prua della nave, e tutti gli scontri avvenivano fra vascelli su rotte parallele o leggermente convergenti.

Ovviamente il calcolatore provvede ad evidenziare l'unità che viene, di volta in volta, presa in considerazione, segnalandone la posizione in modalità inversa, oltre a segnarne la rotta attuale ed il nome che le identifica. A tale proposito, sono costretto a rammaricarmi per l'incompletezza delle mie fonti, che non mi ha consentito di dare un nome ed un comandante ad ognuna delle navi. Quando è stato possibile, infatti le unità sono state identificate, secondo l'uso della marina a vela inglese, con il nome ed il grado e cognome del comandante (così la prima nave dello schieramento inglese è la "Goliath - Captain Foley" e la terz'ultima dello schieramento francese la "Guillame Tell - Capitaine Villeneuve"), tranne che per l'ammiraglia francese ("L'Orient - Amiral Brueys"), il cui capitano di bandiera (ma sotto il controllo dell'ammiraglio francese) era il capitano Casabianca; in alcuni casi, invece la nave è indicata con il suo solo nome e, in, purtroppo, parecchi casi in cui non avevo dati a disposizione, con un numero d'ordine ed il tipo della nave (ad esempio "Dueponti inglese n. 6" o "Fregata francese n. 14").

Il colore di fondo delle scritte identifica, come in Wei-ch'i, l'appartenenza dell'unità ad uno o a l'altro dei due schieramenti. Anche nel punteggio, che appare dopo ogni serie completa di mosse (cioè tutte le navi di entrambe le flotte), la flotta inglese sarà indicata dal punteggio in "reverse" e quella francese dal valore in modalità normale, la flotta controllata dall'utente viene indicata per prima.

Una delle opzioni che il giocatore ha a disposizione con ogni mossa è quella di modificare la rotta della sua nave (R). È necessario, però, tener conto della rotta attuale del vascello; non è infatti consentito fare virate brusche, ad angolo retto o a 180°, poiché le rotte consentite (vengono introdotte tramite il tastierino numerico) sono quelle corrispondenti alle direzioni cardinali ed alle direzioni intermedie principali (Sud-Ovest, Nord-Est ecc.) non sono consentite virate superiori a 45°. Ciò significa che, se la vostra nave sta navigando per Sud, le alternative possibili, oltre a proseguire sulla rotta attuale sono rappresentate da una virata di 45° a babordo (rotta per Sud-Est) o a tribordo (rotta per Sud-Ovest). Premendo il tasto corrispondente alla attuale posizione della nave (5), la nave si mette all'ancora, perdendo così il diritto al turno successivo, ma non per questo la possibilità di venire coinvolta in un duello di artiglierie e di aprire il fuoco.

Nel caso in cui si chieda una correzione di rotta illegale la nave "prende a collo", termine nautico che indica il caso in cui, per una errata manovra, le vele prendono il vento con la faccia anteriore, facendo perdere al vascello l'abbrivio; anche in questo caso la nave è costretta

Abukir 1798 battaglia navale per CBM

a perdere il turno e a ritentare quindi la manovra.

Ovviamente, di tutto ciò bisogna tenere conto quando ci si trova sotto costa, per evitare di andare ad arenarsi sui banchi di sabbia, o quando ci si vuole affiancare ad un vascello nemico per spararglì una bordata.

Un'altra opzione disponibile (tasto S), consente di avere la situazione completa dei danni della propria flotta. Per ogni nave viene segnalata insieme alla sua posizione ed ai dati che la riguardano, l'entità dei danni subiti nel caso in cui la nave sia stata sottoposta ad uno scontro a fuoco (l'entità di questi danni è espressa da una frazione in ragione del limite massimo dei danni che la nave può subire e dei colpi che le sono stati inferti, come vedremo nella parte tattica) - ad esempio, per un dueponti, "ha subito danni per 23/60" -, oppure la sua attuale situazione: "si è arresa", "è disalberata", "non può governare", "è sfuggita", "è affondata", "si è arenata" o (solo per le fregate) "si è incagliata"; o, semplicemente, "non ha subito danni". Vedremo alcune delle voci suddette, ed il loro significato, nella parte "tattica" del gioco; abbiamo già visto in che casi il vascello va considerato "arenato" o "incagliato"; vorrei, qui, semplicemente precisare che una unità deve ritenersi "sfuggita", se esce dai limiti del campo di battaglia.

La funzione ottenibile premendo il tasto V (e che dà il rapporto delle vedette sulle unità avversarie), si può in un certo senso considerare simmetrica rispetto all'opzione descritta nel paragrafo precedente.

Anch'essa infatti segnala se le navi avversarie si sono arenate o arrese o se non sono in grado di governare, ma nel caso in cui non rientrino in nessuna delle voci sopra descritte, invece di segnalare l'entità dei danni subiti, che una vedetta, dall'alto della sua coffa, non sarebbe in grado di valutare con precisione, ne segnala la rotta, informazione, questa, assai utile per avvicinarsi ed attaccare l'unità nemica.

Premendo il tasto X, infine, è possibile salvare su nastro magnetico la situazione della partita in corso, per poterla riprendere in un momento successivo. Non per questo, si deve rinunciare a proseguire immediatamente la partita; a richiesta dell'utente, dopo aver salvato i file necessari, si può proseguire indisturbati verso la più severa delle sconfitte, sicuri di poter riprendere la partita da un punto in cui la propria situazione non era ancora compromessa.

Il calcolatore provvede autonomamente a far rispettare le regole per gli spostamenti strategici; una nave non ha diritto a muovere nei seguenti casi:
a) Nei casi in cui sia da considerarsi "fuori gioco", ovviamente, cioè se è sfuggita o affondata.
b) Se si è arenata o incagliata o se si è arresa in uno scontro precedente.
c) Se salta un turno perché all'ancora o perché ha preso "a collo".
d) Se in uno scontro precedente è stata disalberata (in questo caso non si muove dalla posizione che occupava al momento dello scontro).
e) Se non può governare a causa dei danni subiti. In questo caso la nave non sta però ferma ma procede per la rotta che aveva prima dello scontro, senza alcuna possibilità di modificare la propria situazione: può essere coinvolta in ulteriori duelli di artiglieria, ma non può evitare le secche o anche la altre navi, sia amiche che avversarie.

Da notare che, nel caso in cui due navi si speronino, i danni di ognuno sono proporzionali alla propria stazza ed a quella dell'altra nave, e, nel caso in cui una delle due, o entrambe, abbiano già subito altri danni, può anche portare all'affondamento.

Tattica

Nel caso in cui si siano verificate le condizioni che portano ad uno scontro a fuoco, il calcolatore provvede ad annullare la pagina "strategica", passando a quella tattica.

Su di un mare liscio come l'olio (come in effetti fu, nelle acque riparate della baia, nel corso di tutta la battaglia), compariranno le silhouette dei due vascelli. Per ragioni di ordine pratico le due navi appaiono come se navigassero su rotte parallele anche nel caso in cui si trovino su rotte leggermente convergenti; il computer provvede però a rappresentare la nave avversaria come vista da prua o da poppa a seconda del caso in cui navighi nello stesso verso o in senso opposto rispetto al vostro vascello, che appare sempre come visto da prua. Nel caso in cui uno dei due contendenti si trovi a controllare la batteria francese sull'isola di Abukir, al posto del profilo di un vascello apparirà un castelletto, completo di bandiera, arroccato su di uno scoglio roccioso.

Le navi, sono, ovviamente, proporzionate al loro tipo - le fregate appaiono come dotate di un solo ponte e di due ordini di vele - con i portelli aperti ben in evidenza, mentre poppa e prua possono essere distinte, rispettivamente, per la presenza dello specchio di poppa o dell'asta di bompresso, oltre che, naturalmente, per l'inferitura delle vele.

Le navi inglesi (come detto, tutte dueponti) hanno scafi bianchi e nessuna fiamma sull'albero di maestra, mentre le navi francesi hanno scafi neri ed il tricolore rivoluzionario in testa d'albero. La nave controllata dal giocatore apparirà, sempre, alla sinistra del video, mentre alla destra apparirà quella controllata dal computer; ciò indipendentemente dalla loro posizione relativa e dalla loro appartenenza ad una o l'altra delle due flotte. I nomi delle due navi (con le regole sopra citate per quanto riguarda sia forma che colore delle scritte) appariranno su due righe successive, prima quello della nave controllata dal giocatore, accostato a sinistra, poi quello del vascello avversario, accostato a destra; nelle righe successive, all'altezza della nave

Abukir 1798 battaglia navale per CBM

da voi controllata appariranno i valori dell'alzo e della carica per i cannoni di babordo del vostro vascello. I tasti che permettono di controllare questi due valori fanno parte del tastierino numerico e sono, per la precisione:

4 — Per aumentare l'alzo (valore massimo 45°, dopo di che il tiro da diretto diventerebbe indiretto, ma con valori di gittata ottenibili anche con alzi inferiori ai 45°). Da notare che, al raggiungimento del limite superiore, l'alzo passa al valore minimo.

5 — Per diminuire l'alzo (valore minimo 1°, poiché 0° avrebbe dato dei problemi di congruenza nell'espressione matematica per il calcolo della traiettoria). Sono valide tutte le considerazioni fatte per il caso precedente; lo scarto minimo è di un grado.

N.B. — So che in artiglieria non si utilizzano i gradi sessagesimali, vale a dire quel sistema in cui l'angolo retto misura 90° e ha sottomultipli del grado (primi e secondi) di sessanta in sessanta, bensì i gradi centesimali - sistema in cui l'angolo retto è pari a 100 gradi centesimali -; ma, a parte il fatto che non ho idea di quante persone, al di fuori delle Forze Armate, abbiano confidenza con questo sistema, non disponevo delle funzioni trigonometriche relative.

1 — Per aumentare la carica (la carica massima è rappresentata dall'unità, frazioni successive, fino ad un minimo di mezza carica, hanno valori inferiori di un decimo). Questo valore è stato introdotto per poter controllare la velocità d'uscita del proiettile e quindi la lunghezza del tiro a parità di alzo, l'idea è stata presa dal sistema in voga nella marina a vela inglese, che aveva in dotazione, oltre alla polvere da sparo sfusa, anche cariche confezionate da una, mezza o un quarto di carica, in modo da poter dosare l'esplosivo.

2 — Per diminuire la carica. Valgono per la carica le considerazioni fatte per l'alzo, circa la contiguità dei valori minimo e massimo, in modo da consentire uno scrolling completo e rapido alla ricerca dei valori desiderati.

0 — Per fare fuoco. Quando si ritiene che i valori dell'alzo e della carica, siano quelli voluti - e a patto che non si abbia già una bordata in volo - si può fare fuoco; questa limitazione è stata posta, per impedire di tenere un ritmo di fuoco troppo alto, e perciò scarsamente realistico (i cannonieri dell'epoca dovevano infatti alare a braccia i cannoni fino a che le volate non fossero completamente all'interno dell'opera morta e quindi scovolarli e ricarli dopo ogni bordata).

Nel momento stesso in cui farete fuoco (attenzione a premere bene i tasti voluti, poiché vengono raccolti con una PEEK, non con una GET), apparirà, lungo la fiancata della vostra nave, uno sbuffo di fumo e, quindi la palla di cannone, che inizierà a seguire la sua traiettoria in base ai dati da voi introdotti ed a quelli legati al tipo della nave.

Quest'ultimi riguardano la velocità di uscita del proiettile, legata, come detto alla carica, ma anche alle sue dimensioni; a questo scopo si è assunto che la velocità di uscita dei proiettili è pari a 100 metri al secondo per ognuno dei ponti della nave (per cui una fregata spara proiettili con una velocità iniziale di 100 m/sec., un dueponti 200 e l'ammiraglia francese 300 m/sec.), questo nel caso in cui la carica sia pari a 1, ovviamente valori proporzionalmente inferiori si ottengono con frazioni di carica. Le palle di cannone delle due navi si distinguono per il loro colore, nero per le francesi e bianco per le inglesi.

Il tiro prosegue fino a che non si presenta uno dei seguenti casi:

a) Il tiro risulta troppo corto e la palla finisce in acqua. In questo caso si alza una colonna d'acqua dove il colpo è caduto e si può far partire una nuova bordata.

b) Le due palle, nelle rispettive traiettorie, si incontrano in uno stesso punto-video. Quella delle due che ha colpito l'altra si disintegra e l'autore della bordata può spararne un'altra.

c) La palla esce dai limiti dello schermo, o superiormente o oltre la nave avversaria senza averla però colpita. Subito dopo la sua scomparsa è possibile far di nuovo fuoco.

d) Il colpo va a segno. In questo caso comparirà sullo schermo, in corrispondenza del punto colpito un carattere che, nel caso dello scafo, dell'alberatura e delle parti estreme delle vele è una croce di S. Andrea (dovrebbe rappresentare le schegge di legno spezzate); mentre, se il colpo colpisce di netto le vele, in corrispondenza apparirà un buco. Dopo di che si può far partire un'altro colpo.

Ovviamente l'efficacia dei colpi è funzione del loro peso (e quindi della nave che li fa partire, navi più grossa avranno armamenti più potenti), ed anche del punto che si colpisce. I valori, cui va aggiunta una frazione decimale "random" per rendere il gioco più vivace, sono in ragione di 1 punto per ogni ponte della nave che ha fatto fuoco, nel caso in cui si colpisca l'alberatura o le vele; e di tre punti - sempre per ogni ponte se si colpisce lo scafo; per cui un colpo a segno di una fregata all'attrezzatura di un dueponti inglese vale un solo punto, un colpo allo scafo da parte di un dueponti vale 6 punti ed uno de "L'Orient" addirittura 9.

Lo scontro prosegue fino a che non si verifica una delle seguenti evenienze:

a) Una delle navi si arrende (in genere per evitare di essere affondata e se non ha speranze residue di affondare l'avversaria). Il giocatore può farlo premendo il tasto "?", nel qual caso apparirà in testa l'albero alla sua nave una bandiera bianca. Vedremo poi come, a volte, risulti più conveniente la resa.

b) Entrambe le navi esauriscono i colpi (venti a testa per ogni ponte). Capita molto raramente (che io ricordi una sola volta); comunque in scontri successivi la nave recupera la

Abukir 1798 battaglia navale per CBM

sua dotazione di colpi (sulle navi era compito dell'armaiolo confezionare nuove cariche se necessario).
c) Una delle navi viene affondata. Perché ciò avvenga l'avversario (o più avversari, se ha partecipato a più scontri), deve averle inferto un numero di colpi tale che, per gravità, essi superino il bonus a disposizione della nave. Questo bonus è fissato, come al solito, in base all'indice più immediato della stazza della nave, vale a dire il numero dei suoi ponti; e, per essere più precisi, in ragione di venti punti per ogni ponte (20 per una fregata, 40 per un dueponti, 60 per un treponti), indipendentemente da dove essi vengono inferti. Ovviamente, a causa della parte decimale "random", non è possibile precisare esattamente quanti colpi siano strettamente necessari per affondare ogni nave di ciascun tipo.

Esistono però altre possibilità:
a) Una delle navi (o anche entrambe) riceve una quantità di colpi all'alberatura la cui gravità è pari o superiore alla metà dei colpi sufficienti per affondarla; in questo caso la nave, nel corso dello scontro, viene disalberata (al posto dell'alberatura crollata, viene fornito un albero di fortuna) e, se sopravvive allo scontro, non può più abbandonare la posizione che occupava prima del duello.
b) Una delle due navi sopravvive allo scontro ma dopo aver subito danni pari o superiore ai nove decimi di quelli necessari al suo affondamento, in questo caso la nave non è più in grado di governare ed è costretta a proseguire sulla rotta che aveva prima dello scontro, con tutto ciò che la cosa comporta (probabilità di arenarsi od urtare altre navi, la qual cosa porterebbe, molto probabilmente, al suo affondamento).

N.B. — Ovviamente le navi disalberate o non in grado di governare non perdono il diritto di sparare e di farsi sparare addosso come tutte le altre!

Nel caso della batteria francese sull'isola non si può parlare di affondamento o disalberamento e, tantomeno, di impossibilità di governare, i colpi inferti sull'isola, ovviamente, non hanno alcun valore, mentre quelli che colpiscono il forte hanno coefficienti di efficacia pari a quelli allo scafo di una nave (cioè 3; poiché le navi inglesi sono tutte dei dueponti, e solo gli inglesi sono soggetti al tiro della batteria, ogni colpo di una nave inglese che colpisce il

PERSONAL SOFTWARE

Abukir 1798 battaglia navale per CBM

forte vale 3 x 2 = 6 punti), le schegge di pietra dei parapetti erano infatti altrettanto micidiali dei colpi stessi: mentre il peso dei colpi sparati dalle batterie del forte è di 5 (come se il forte fosse cioè un ipotetico mastodonte a cinque ponti), le batterie a terra, infatti, sparavano generalmente a colpi arroventati in una fornace, in modo da incendiare il legname di cui le navi a vela erano fatte; per cui, in genere, quattro o cinque colpi all'alberatura (5 punti certi, più il fattore casuale), bastano a disalberare un dueponti; ed un paio di colpi allo scafo (15 punti!) in più ad affondarlo. Con queste premesse, è abbastanza naturale affermare che è necessario fare molta attenzione a non capitare a tiro della batteria francese e, se dovesse succedere, è conveniente arrendersi subito.

Come gioca il calcolatore

Come detto, il computer utilizza una routine molto più rapida che in '14-'18, grazie anche al fatto che in Abukir 1798, sono presenti meno opzioni disponibili ad ogni mossa. In pratica i meccanismi di comportamento sono per entrambe le sezioni in cui il gioco è suddiviso, assai semplici. Le prestazioni del calcolatore sono molto buone dal punto di vista strategico mentre appena discrete per quanto riguarda l'aspetto tattico.

Nel primo caso il computer deve provvedere a gestire il cambiamento di rotta o la permanenza sulla rotta attuale. Nel caso in cui controlli la flotta inglese dovrà, ovviamente, badare a non portarsi a tiro delle micidiali batterie a terra. Per entrambe le flotte, invece, si tratterà di evitare speronamenti di navi amiche, arenamenti e, ovviamente gli scontri con vascelli le cui condizioni sono migliori delle proprie o, a parità di condizioni, le cui dimensioni siano maggiori. Ciò fa si che anche una semplice fregata può attaccare un dueponti, se quest'ultimo è sufficientemente danneggiato da far pensare che qualche colpo ben piazzato possa farlo affondare (tra l'altro le piccole fregate sono ingannevolmente difficili da colpire!).

Da un punto di vista tattico andiamo un po' meno bene; il meccanismo di determinazione dell'alzo e della carica verrà descritto nei REMarks, ciò che qui conta sottolineare è che, ovviamente, il calcolatore è in grado di calcolare esattamente la traiettoria dei proiettili, condizione prima, quest'ultima, per poter far fuoco con la massima precisione; i problemi derivano dai meccanismi di casualità che si applicano sia ai cannoni controllati dal calcolatore, sia a quelli controllati dal giocatore. Questi meccanismi, tendono ad esprimere le imperfezioni delle armi dell'epoca (le irregolarità nelle anime dei cannoni, la non perfetta sfericità dei colpi ecc.) oltre agli errori umani, presenti in ogni epoca. Ciò fa si che le percentuali di tiro del computer siano leggermente inferiori a quelle di un giocatore ben allenato (per esempio... mio fratello); in sintesi il computer tende a colpire un po' troppo spesso le vele (pochi punti!) e non è esente da errori (colpi corti in acqua).

Nel corso degli scontri, inoltre, il computer ha messo in luce una preoccupante mancanza di combattività, che lo portava ad arrendersi non appena era ad un passo dall'affondamento e nell'assoluta impossibilità di riuscire ad affondare per primo l'avversario, ciò a causa del fatto che, in base alle condizioni di vittoria stabilite per il gioco, è più conveniente avere una nave danneggiata che si arrende al nemico che una nave di coraggiosi testardi in fondo alla baia di Abukir.

Le condizioni di vittoria

Le condizioni di vittoria sono determinate in base alla situazione di tutte le unità che compongono le due flotte. Ovviamente il peso maggiore è dato dalle navi che si è riusciti ad affondare: si ottengono 30 punti per ogni ponte delle navi che si è affondato (è per questo che, nell'esprimere i danni, essi vengono dati come frazioni di 30-esimi per le fregate, 60-esimi per i dueponti e 90-esimi per la nave ammiraglia francese). A ciò va sommato il contributo dato dai danni subiti dalle navi che non sono affondate, e che sono quindi comprensivi di eventuali speronamenti, arenamenti su banchi di sabbia ed incagliamenti, e l'eventuale contributo, per la sola flotta francese, dei danni subiti dal forte.

È per questa ragione che può essere conveniente arrendersi: la resa di un dueponti poco prima di affondare può costare circa 35-38 punti - con più di quaranta affonderebbe -, ma la nave non può più essere coinvolta (a meno che non venga speronata!) e quindi si evitano i 60 punti dell'affondamento.

Le navi sfuggite, cioè uscite dal campo di battaglia, non contano ai fini del punteggio e, quindi, a volte può essere conveniente anche darsela a gambe.

Il punteggio massimo acquisibile è di 720 punti per la flotta francese, e di 940 per quella inglese e le condizioni di vittoria sono state fissate di conseguenza: è possibile vincere se si supera per primi i cinquecento punti con almeno cento punti di vantaggio sull'avversario, o i seicento punti con almeno cinquanta di vantaggio, o, infine, se si supera per primi i 700 punti. Oltre a queste condizioni di vittoria, si debbono aggiungere le logiche condizioni di terminazione: se una delle due flotte non ha più vascelli in ordine di combattimento, perché affondati o arenati o fuggiti od arresi, la partita ha ovviamente termine, ma non necessariamente con la vittoria dell'avversario, se infatti la flotta scomparsa ha inflitto all'avversario più danni di quanti ne abbia subiti (ad esempio perché molte delle navi si sono date alla fuga), essa può aggiudicarsi la partita.

Ultimi consigli

Ai principianti, consiglierei di far patica controllando la più potente flotta francese, prima di azzardarsi ad affrontare con i soli dueponti inglesi i cannoni de "L'Orient" e degli altri sedici vascelli francesi. Per quanto riguarda il comportamento tattico è ovviamente preferibile colpire basso, allo scafo, anche correndo qualche rischio di vedere i propri colpi finire in acqua, poiché è molto più redditizio. ■

(Continua)

(Per ragioni di spazio pubblicheremo il listato e i REMarks sulla prossima puntata).

Dal BASIC al Pascal

— Parte terza —

Strutture di controllo dei cicli e delle diramazioni

a cura della *Redazione*

Strutture di controllo dei cicli

Nel BASIC, ci sono due strutture di ramificazione e una struttura esplicita di ciclo. Le strutture di ramificazione sono IF...THEN...ELSE e ON N GOTO. Ecco alcuni esempi:

```
100 if a=b then print "uguali"
    else print "diversi"
200 if a=-2 then n=1
210 if a=1 then n=2
220 if a=5 then n=3
230 rem
240 on n.goto 300,400,500
```

La struttura IF...THEN...ELSE dovrebbe essere nota a chiunque usi il BASIC, mentre la ON N GOTO non è impiegata largamente perché scomoda. Il GOTO è seguito da una lista di numeri di righe. Se la variabile (che qui chiamiamo N) ha il valore 1, il primo numero di riga è la destinazione del salto. Se la variabile ha il valore 2, il secondo ecc. I diversi tipi di BASIC agiscono in vari modi nel caso in cui la variabile abbia un valore fuori dell'intervallo. Alcuni saltano alla riga seguente la riga ON N, altri al primo numero di riga dato; ecc. Consultate il vostro manuale di BASIC. La difficoltà qui consiste nella lista di dichiarazioni che deve precedere la ON N se il salto deve essere fatto secondo un certo insieme di decisioni logiche.

Nel Pascal, l'equivalente della dichiarazione IF...THEN...ELSE è la dichiarazione IF...THEN...ELSE!

Sì, sono identiche! O quasi. Il Pascal permette l'uso di dichiarazioni composte, come abbiamo già visto.

```
IF A=-2
  THEN
    BEGIN
      P:=17;
      Q:=69;
    END
  ELSE
    BEGIN
      P:=69;
      Q:=17;
    END;
```

Questo è un esempio semplificato, ma l'idea è chiara. Notate che la IF... THEN...ELSE è una dichiarazione unica, di cui entrambe le parti possono essere composte. È importante che non ci sia un punto e virgola dopo la parte THEN della dichiarazione, poiché la ELSE fa parte della stessa dichiarazione. All'interno della dichiarazione composta, però, i punto e virgola sono richiesti per separare le dichiarazioni multiple. Notate che la parte ELSE è facoltativa così come lo è nel BASIC.

L'equivalente nel Pascal della ON N GOTO è la dichiarazione CASE. Questa struttura è nota come "salto ad N vie". L'equivalente CASE dell'esempio BASIC di prima è:

```
CASE A OF
  -2:NEGATIVO;
   1:UNITA;
   2:GRANDE
END;
```

Gli identificatori NEGATIVO, UNITA' e GRANDE sono delle procedure da eseguire in ognuno dei tre casi. Dopo l'esecuzione della procedura scelta, il programma continua alla riga dopo la dichiarazione CASE. Il Pascal standard non prevede la situazione "nessuno di queste". Se non è soddisfatta alcuna delle condizioni previste dalla dichiarazione CASE, il Pascal riporta un errore. Alcuni sistemi hanno una clausola ELSE facoltativa alla fine della CASE per evitare gli errori. Altri sistemi usano la parola OTHERWISE (altrimenti). Se avete una versione standard, l'unico modo di evitare un "system error message" che confonde completamente la persona che adopera il programma senza averlo scritto, è di controllare per vedere se una delle condizioni è soddisfatta e, se no, dare un messaggio di errore che ha senso per chi usa il programma. In questo caso, si potrebbe usare una dichiarazione IN per il controllo:

```
IF A IN [-2,1,5]
  THEN
    BEGIN
      CASE A OF
        -2:NEGATIVO;
         1:UNITA;
         5:GRANDE
    END;
  ELSE
    WRITE('MESSAGGIO DI ERRORE');
```

Il Pascal permette con l'istruzione CASE la stessa flessibilità che permette con gli indici di una matrice. Funziona sia con variabili CHAR che con tipi scalari. Ed in effetti è molto utile con i tipi scalari, perché, sebbene una variabile del tipo GIORNO DELLA SETTIMANA (GIORNO per esempio) possa assumere i valori DOM, LUN, MAR... (vedete l'esempio di prima), non è possibile usare una dichiarazione stampa come WRITE (GIORNO). Bisogna usare una dichiarazione CASE:

```
CASE GIORNO OF
  LUN:WRITE('LUNEDI');
  MAR:WRITE('MARTEDI');
  MER:WRITE('MERCOLEDI');
  GIO:WRITE('GIOVEDI');
  VEN:WRITE('VENERDI');
  SAB,DOM:WRITE('WEEKEND')
END;
```

Dal BASIC al Pascal

Un altro esempio con l'astensione OTHERWISE disponibile in alcuni Pascal:

```
IF LETTERA IN ['A'..'Z"]
  THEN
    BEGIN
      CASE LETTER OF
        'A','E','I','O','U':WRITE('VOCALE');
        'S','T','N':WRITE('FREQUENTE')
      END;
      OTHERWISE WRITE('ORDINARIA');
    END;
```

Notate che qui non è necessario chiamare una procedura. La condizione CASE può essere seguita da una dichiarazione, anche se composta. Quindi vedete, la dichiarazione CASE è abbastanza potente nel Pascal.

Il ciclo

Il BASIC ha una struttura di controllo dei cicli. Non c'è bisogno che diamo qui un esempio di un ciclo FOR-NEXT. Il Pascal ha un equivalente quasi uguale:

```
FOR I:=1 TO 10 DO
  ARRAY[I]:=0
FOR N:=1 TO 15 DO
  BEGIN
    REALVAL:=REALVAL+N/10;
    WRITE(REALVAL)
  END;
```

Notate che non è richiesto alcun NEXT. La fine della frase segna la fine del ciclo. Questo naturalmente è possibile a causa della caratteristica del Pascal di permettere dichiarazioni composte. Il Pascal non permette una specificazione STEP come il BASIC. Il motivo è che l'indice del ciclo è un numero intero, il che rende impossibili i valori frazionari. Un ciclo con un passo diverso dall'unità si fa meglio in un altro modo, a causa dell'inesattezza di somme e differenze ripetute nella matematica a virgola mobile.

Lo si può fare in un ciclo FOR-DO (e questa è una pratica migliore anche nel BASIC), secondo lo schema mostrato nell'esempio. Il Pascal può fare i passi anche all'indietro, cioè può usare un passo di un'unità negativa, usando la parola DOWNTO anziché TO. Per esempio:

FOR K: = 10 DOWNTO 1 DO...

I numeri negativi interi sono permessi come indici ma non come numeri reali. La variabile indice deve essere stata dichiarata prima. Sebbene non siano permessi indici reali, sono permessi altri tipi di indici, precisamente CHAR e scalari.

```
FOR GIORNO:=LUN TO DOM DO...
FOR LETTERA:='A' TO 'B' DO...
```

Il Pascal ha di più

Qui di nuovo, vedete più flessibilità, alle spese di maggior complessità. Nel Pascal, sono permessi altri due tipi di strutture di ciclo. Il BASIC non ha qualcosa di direttamente paragonabile, ma le seguenti strutture possono essere simulate per mezzo di una dichiarazione GOTO nel BASIC. Queste strutture sono la WHILE-DO, e la REPEAT-UNTIL.

```
WHILE A<B DO
  BEGIN
    A:=A+0.15;
  END;

REPEAT
  A:=A+0.15;
UNTIL A>=B;
```

C'è una differenza fra le due. Nella struttura WHILE-DO, si controlla la condizione all'inizio del ciclo. Se essa è già soddisfatta, il ciclo non viene eseguito (neanche una volta) e l'esecuzione continua dopo il ciclo. Nella struttura REPEAT-UNTIL, d'altra parte, il ciclo si esegue prima che la condizione venga controllata. Anche se la condizione è vera prima di arrivare al ciclo, le istruzioni nel ciclo sono eseguite almeno una volta. La differenza è sottile, ma utile. Per divertirci, ecco come si farebbe la stessa cosa nel BASIC:

```
100 if a<b then 110 else 200
110 rem istruzioni
120 a=a+.15
130 goto 100
200 rem il programma continua qui
```

```
100 rem istruzioni
110 a=a+.15
120 if a>= then 200 else 100
200 rem il programma continua qui
```

Questo è il primo esempio di come si impara il BASIC con il Pascal. Abbiamo visto un paio di strutture utili nel Pascal e mostrato come simularle nel BASIC.

Notate che in tutti gli esempi sopracitati, è estremamente importante che la condizione controllata nel ciclo venga modificata in qualche punto all'interno del ciclo in modo che prima o poi diventi vera.

(L'eccezione esiste se fate andare il Pascal su un computer che permette l'esecuzione parallela).

```
WHILE A<>B DO
BEGIN
  A:=A+1;
  (ISTRUZIONI)
END;
```

Questo ciclo si compie 32766 volte e poi dà un messaggio di INTEGER OVERFLOW se all'inizio si hanno B = 1 e A = 2. Se B = 1,5 all'inizio (che potrebbe succedere solo se fosse reale), il ciclo continua per sempre. ■

Impariamo il linguaggio macchina con il VIC e C 64

— Parte prima —

Prima puntata di una interessante serie di articoli

di *Alessandro Guida*

Parallelamente agli articoli riguardanti il linguaggio macchina per ZX81 e Spectrum, che come è noto utilizzano microprocessore Z80, ci sembra giusto parlare anche del micro 6502 molto diffuso, e adottato, oltre che dai prodotti di casa Commodore anche da altri popolari personal, primo fra tutti Apple II.

Introduzione

L'intenzione di questa serie di articoli è insegnare in maniera semplice, gli elementi più importanti della programmazione in linguaggio macchina.

Seguiremo una strada abbastanza insolita. Useremo, fin dove è possibile, il BASIC per facilitare l'apprendimento dei principi fondamentali. Questo perché, essendo il BASIC un linguaggio ad alto livello, permette una programmazione di tipo più descrittiva che non il veloce, ma estremamente sintetico, linguaggio macchina. Daremo, quindi, per scontata la vostra conoscenza del BASIC e procederemo per gradi dal linguaggio più evoluto al linguaggio proprio del computer.

Perché il linguaggio macchina

Il pregio fondamentale del linguaggio macchina è senz'altro la sua velocità. Si possono ottenere incrementi di velocità tra un'operazione svolta in BASIC e la stessa scritta in linguaggio macchina, anche superiori alle 100 volte. Inoltre ci sono casi in cui la programmazione in Assembler (lo strumento che consente di programmare il computer in linguaggio macchina) è più agevole, o addirittura indispensabile. Ne vedremo diversi esempi.

```
   01001011

 0x128=   0
 1x64  = 64
 0x32  =  0
 0x16  =  0
 1x8   =  8
 0x4   =  0
 1x2   =  2
 1x1   =  1
 -----------
         75
```

Figura 1. *Esempio di conversione binario decimale.*

Cos'è il linguaggio macchina

Finora avete programmato il vostro computer in un linguaggio che è chiamato BASIC. Probabilmente sapete anche che all'interno del vostro computer c'è un microprocessore (il circuito integrato che esegue le varie operazioni programmate) che per il VIC ha la sigla 6502 mentre per il 64 la sigla 6510. Quello che invece vi stupirà è che le due cose, il BASIC e il Microprocessore, sono incompatibili uno con l'altro. Ossia, il BASIC (così come lo battete da tastiera) non sa come parlare al microprocessore, e quest'ultimo non capisce quello che gli viene comunicato in BASIC. In pratica sono come due stranieri che parlano lingue diverse. Il problema è che il 6502 (6510) è in grado di ricevere ed eseguire solo istruzioni in linguaggio macchina.

Queste istruzioni sono formate da un codice, che altro non è che un numero compreso tra 0 e 255. Tenendo conto del fatto che alcuni codici non corrispondono ad alcun comando si vede che le operazioni che il microprocessore è in grado di svolgere da solo sono molto poche (circa 150). Inoltre, queste operazioni, che in seguito descriveremo, una per una, sono molto semplici e nulla hanno a che vedere con gli statement a volte complessi del BASIC. Perciò il BASIC è dotato di un "interprete" che traduce ogni singolo comando BASIC in una serie di istruzioni in linguaggio macchina. Quindi, ogni volta che viene incontrato un certo statement (comando BASIC) questo viene confrontato con una lista di comandi che l'interprete è in grado di capire e di conseguenza interpretato e, poi, eseguito.

Se provate a pensare che questo iter è seguito ogni volta, anche se si tratta sempre dello stesso comando, capirete il perché della lentezza del BASIC, e il motivo per cui conviene scrivere le routine più impegnative di un programma in linguaggio macchina.

Prima di andare avanti, voglio sottolineare che anche se il microprocessore del VIC (6502) e quello del 64 (6510) hanno sigle diverse ai fini della programmazione sono assolutamente identici. Quindi, d'ora in avanti, ci riferiremo solo al 6502 ben sapendo che quanto verrà detto sarà valido anche per il 6510.

Date queste piccole premesse possiamo incominciare lo studio del linguaggio del microprocessore 6502.

Decimale	Binario	Decimale	Binario
0	00000000	32	00100000
1	00000001	33	00100001
2	00000010	=	=
3	00000011	=	=
4	00000100	63	00111111
5	00000101	64	01000000
6	00000110	65	01000001
7	00000111	=	=
8	00001000	=	=
9	00001001	127	01111111
10	00001010	128	10000000
11	00001011	129	10000001
12	00001100	=	=
13	00001101	=	=
14	00001110	=	=
15	00001111	253	11111100
16	00010000	254	11111110
17	00010001	255	11111111
=	=		
=	=		
31	00011111		

Figura 2. *Tabella dei numeri binari minori di 255.*

Decimale	Binario
+127	01111111
+126	01111110
=	=
+64	01000000
+63	00111111
=	=
+32	00100000
+31	00011111
=	=
+16	00010000
+15	00001111
=	=
+8	00001000
+7	00000111
=	=
+4	00000100
+3	00000011
+2	00000010
+1	00000001
0	00000000
-1	11111111
-2	11111110
-3	11111101
-4	11111100
=	=
-8	11111000
-9	11110111
=	=
-16	11110000
-17	11101111
=	=
-32	11100000
-33	11011111
=	=
-64	11000000
-65	10111111
=	=
-127	10000001
-128	10000000

Figura 3. *Tabella dei numeri binari relativi in complemento a 2.*

La notazione binaria

Poiché il nostro microprocessore è nato per trattare dei numeri è giusto vedere, subito, in quale forma sono rappresentati al suo interno.

La rappresentazione dei numeri da noi normalmente utilizzata è detta decimale poiché si tratta di un tipo di numerazione in base 10 in cui ogni cifra può assumere un valore da 0 a 9.

In altre parole un numero come 1532 lo possiamo scrivere come: $1x10^3 + 5x10^2 + 3x10^1 + 2x10^0$.

E infatti avremo 1x1000 + 5x100 + 3x10 + 2x1 = 1532.

La notazione binaria, invece, è in base 2. Ciò vuol dire che ogni cifra è moltiplicata per una potenza di 2 e può assumere come valore solo 0 o 1.

Ad esempio il numero 133 sarà rappresentato come:
1x2 7+0x2 6+0x2 5+0x2 4+0x2 3+1x2 2+0x2 1+1x2 0.

Così 00000001 in binario equivale a 1 in decimale, 00000010 equivale a 2, 00000011 è uguale a 3 e così via.

In pratica per trasformare un numero binario in decimale sarà sufficiente sommare il valore delle cifre in cui compare l'uno tenendo presente che tali valori sono le corrispondenti potenze di 2, quindi ognuno sarà il doppio del precedente (da destra a sinistra).

Gli esempi che abbiamo portato non a caso utilizzano le potenze di 2 da 0 a 7. Infatti i dati in ingresso o uscita dal microprocessore possono avere come valore massimo 255, cioè 11111111 in binario.

Importante: un numero, compreso tra 0 e 255, utilizzato dal 6502 è detto byte. Ognuna delle otto cifre che compongono un byte si dice bit. Questi bit sono numerati secondo le corrispondenti potenze di 2, vanno cioè dal bit 0 al bit 7.

Riassumento: il computer può gestire numeri, detti byte, formati da 8 bit. Questo limita il campo a valori compresi tra 0 e 255, in decimale.

Impariamo il linguaggio macchina con il VIC e C 64

Decimale	Esadecimale	Decimale	Esadecimale
0	0	16	10
1	1	32	20
=	=	48	30
=	=	64	40
8	8	80	50
9	9	96	60
10	A	=	=
11	B	=	=
12	C	160	A0
13	D	=	=
14	E	=	=
15	F	240	F0

Figura 4. *Tabella di conversione da decimale ad esadecimale.*

Ogni bit può essere uguale a 1 o 0. L'uno è detto stato alto, lo zero stato basso. Il bit 0 è il primo a destra ed è detto LSB (bit meno significativo), il bit 7 è il primo a sinistra ed è chiamato MSB (bit più significativo).

Stabilito ciò, il successivo passo da compiere è definire l'operazione più importante: l'addizione.

Per sommare due numeri binari ci si comporta come facciamo normalmente nel sistema decimale. Quindi, si sommeranno le cifre (i bit) corrispondenti dei due numeri, tenendo presente che il valore più alto ammesso è uno.
Abbiamo, cioè, quattro possibilità:

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0 e riporto 1.

Vediamo alcuni esempi di addizione:

```
(9) 00001001 +
(2) 00000010 =
    --------
    00001011 (8+4+2=11
              risultato esatto)

(15)00001111 +
 (1)00000001 =
    --------
    00010000 (16 risultato esatto)
```

N.B. Da notare in questo esempio il riporto dal bit 0 al 1, al 2 fino al 4
Un ultimo esempio:

```
(129) 10000001 +
  (1) 00000001 =
      --------
      10000010 (128+2=130
                va bene)
```

Consideriamo ora questo esempio:

```
(130) 10000010 +
(128) 10000000 =
      --------
      00000010 (2 risultato errato)
```

Il risultato atteso era 258. Quindi se la somma dei due numeri supera 255 il risultato sarà errato. Per risolvere questo problema nel microprocessore è inserito il flag C (Carry). Un flag è un registro ad un solo bit, che, quindi, può essere solo attivato (=1) o disattivato.
Il flag C sarà posto uguale a 1 ogni volta che si avrà un riporto oltre l'ottavo bit. In pratica funziona come un nono bit. Riprendendo l'esempio avremo:

```
  (130) 10000010 +
  (128) 10000000 =
        --------
C=1     00000010 (256+2=258
                  risultato esatto)
```

```
 2A4F

  2 x 4096 = 8192
 10 x  256 = 2560
  4 x   16 =   64
 15 x    1 =   15
 ----------------
            10831
```

Figura 5. *Esempio di conversione esadecimale-decimale.*

Prima di proseguire, abbiamo bisogno di una nuova definizione. Definiremo *complemento di un bit* il suo opposto. Quindi il complemento di 1 sarà 0, e quello di 0 sarà 1.

Finora abbiamo usato byte con valori positivi compresi tra 0 e 255. È invece auspicabile disporre anche di numeri negativi. Il sistema utilizzato dal 6502 per rappresentare i numeri minori di 0 e chiamato 'complemento a due'.
Infatti, se prendiamo un numero positivo per ottenere il suo negativo dovremo complementare ogni bit e infine aggiungere 1.
Se ad eempio vogliamo −67 in binario scriveremo:

```
01000011  (+67)
10111100  (lo complementiamo)
10111101  (aggiungiamo 1)
```

Quindi la rappresentazione binaria di −67 è 10111101.
La rappresentazione dei numeri negativi appena vista, ha due grandi pregi.
1 - Un numero negativo può essere facilmente riconoscibile dal bit 7 uguale a 1.
2 - Fornisce risultati esatti nella somma tra numeri relativi.
Infatti:

```
    (18) 00010010 +
   (−11) 11110101 =
         --------
C=1      00000111 (4+3+1=7
                   risultato esatto
```

N.B. Per ora ignoriamo il carry.
Sicuramente avrete già capito che questo è, anche, il metodo per sottrarre due numeri. Cioè si somma al primo il complemento a due del secondo.

Impariamo il linguaggio macchina con il VIC e C 64

Ancora un esempio, vogliamo calcolare 24-36:

```
  (24)  00011000 +
 (-36)  11011100 =
        --------
  C=0   11110100 (-12
                 risultato esatto)
```

È facile capire che, in effetti, si tratta di un numero negativo poiché il bit 7 è =1. Ma controlliamo che il suo valore assoluto sia davvero 12.

```
11110100  (è il risultato negativo)
00001011  (lo complementiamo)
00001010  (togliamo 1)
```

Abbiamo così il risultato in positivo: 12, come sperato.
Ma nell'uso della rappresentazione in complemento 2 esiste un problema. Osservate questo esempio:

```
 (64)  01000000 +
 (65)  01000001 =
       --------
 C=0   10000001 (-127
                risultato errato)
```

Avremmo dovuto ottenere 129. Il problema nasce dal fatto che con l'introduzione dei numeri relativi il campo non va più da 0 a 255 ma da −128 a +127. Così se nella nostra somma si ha un riporto dal bit 6 al bit 7 si cadrà in errore, poiché il bit 7 è il bit del segno.
Un altro esempio:

```
 (- 63)  11000001 +
 (-125)  10000011 =
         --------
  C=1    01000100 (+68
                  risultato errato)
```

Ora si è avuto un riporto, ma il bit 7 è stato lo stesso cambiato accidentalmente, causando un risultato non esatto.

Abbiamo quindi bisogno di un nuovo flag che ci indichi quando il bit 7 è stato variato per errore.
Questo flag chiamato Overflow è già previsto all'interno del 6502, ha nome V, e viene settato ogni volta che si ha un riporto dal bit 6 al 7 oppure dal bit 7 al carry.
Ciò si verifica, in pratica, quando si eseguono operazioni tra numeri in valore assoluto molto grandi.

Bisogna specificare che il flag di overflow non sempre indica un errore. Se, infatti, si svolgono elaborazioni su numeri solo positivi il flag di overflow va ignorato non essendo condizione di errore il cambiamento del settimo bit.
Vedremo che invece andrà controllato il flag di carry.

Spesso è necessario operare con numeri più grandi di 255 o fuori dall'intervallo +127, −128. In questi casi si possono utilizzare due o più byte consecutivi.
Normalmente ne utilizzeremo 2. Il bit 0 del secondo byte diventerà così, il bit 8 dell'intero numero.
Il secondo byte si dice (come per i bit) byte più significativo
Poiché 2 elevato a 16 da 65536 con due byte avremo a disposizione tutti i numeri compresi tra 0 e 65535, o tra +32767 e −32768.
Per esempio, il numero 1024 lo scriveremo:

```
         00000100 00000000

bit   15        8 7        0
byte      + sign.   - sign.
```

Nel trasformare un numero binario a 16 bit in decimale è sufficiente leggere il valore degli 8 bit più significativi, moltiplicarlo per 256 e aggiungervi il valore degli altri 8.
Un altro esempio, rappresentiamo il numero 15360:

```
   00111100     00000
```

La numerazione esadecimale

Naturalmente la rappresentazione binaria ha il difetto di non essere per niente pratica. Soprattutto se usiamo numeri a 16 bit. Per ovviare, a questo inconveniente, generalmente si utilizza la notazione esadecimale. Il numero binario viene diviso in gruppi di 4 bit. Con 4 bit si coprono tutti i valori da 0 a 15 (16 numeri da cui il nome esadecimale). I primi 10 saranno le cifre da 0 a 9 seguite dalle lettere da A a F.
Ecco alcuni esempi:

decimale	67	
binario	0100	0011
esadec.	4	3
decimale	161	
binario	1010	0001
esadec.	A	1
decimale	255	
binario	1111	1111
esadec.	F	F

I gruppi di 4 bit si dicono nibble. Ogni nibble rappresenta una potenza di 16.
La conversione da esadecimale in decimale è altrettanto semplice:

1A5F = $1 \times 16^3 + 10 \times 16^2 + 5 \times 16^1 + 15 \times 16^0$ = 1x4096 + 10x256 + 5x16 + 15x1 = 6571.

Il programma

Spero che abbiate seguito fin qui, essendo queste premesse essenziali per la comprensione degli articoli futuri. Il listato 1 è un programmino che esegue le conversioni da decimale, esadecimale o binario in una qualsiasi delle tre notazioni. Potete utilizzarlo anche per eseguire degli esercizi eseguendo le conversioni da voi e, poi, confrontando i vostri risultati con la risposta del computer.
Per l'uso è sufficiente introdurre il dato preceduto da:

D—per decimale
H—per esadecimale
B—per binario

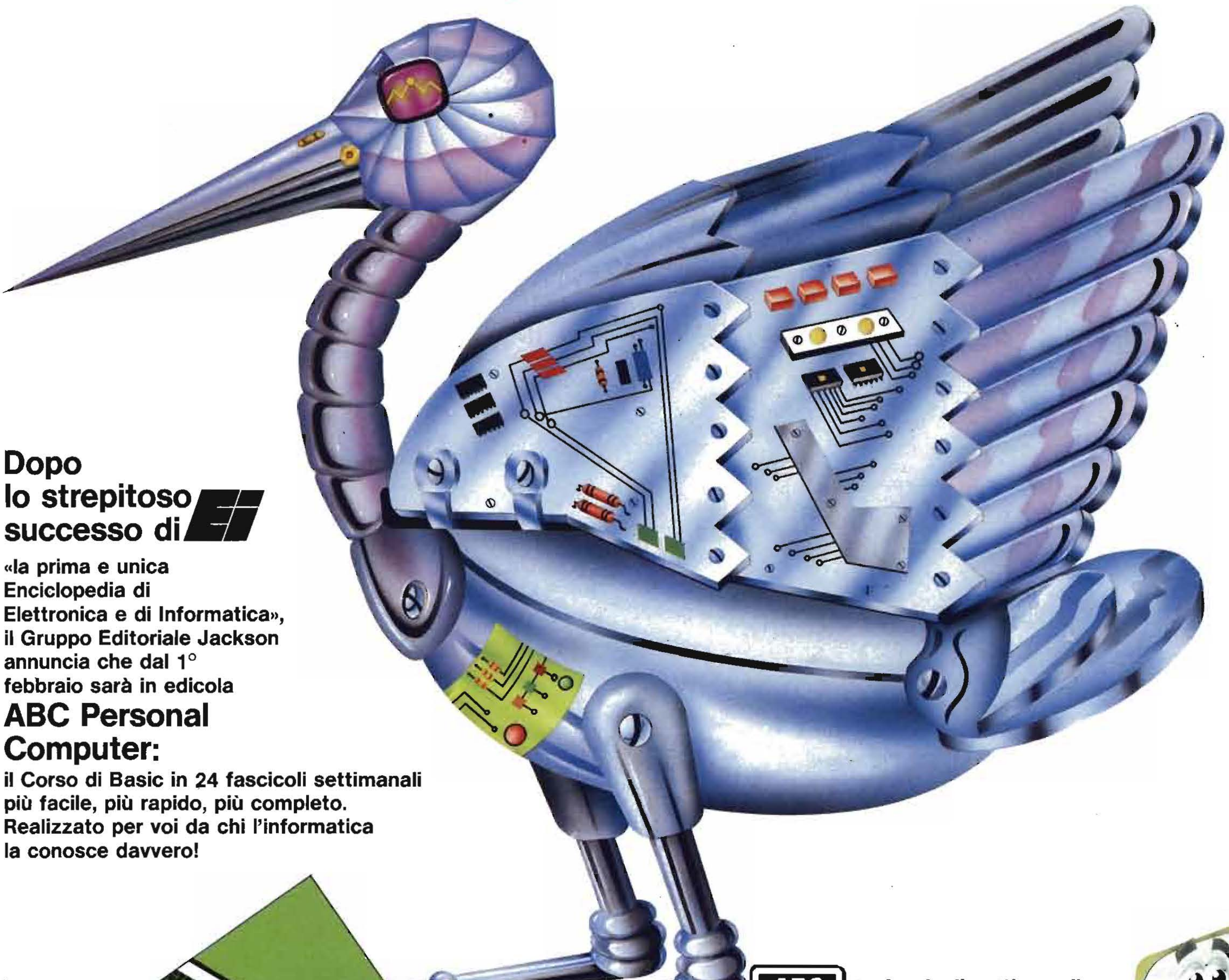
Terminate il vostro numero con il RETURN seguito da una delle lettere viste prima per indicare il formato in cui lo volete tradurre. ■

```
10 PRINT:PRINT "D-Decimala":PRINT "B-Bin
ario":PRINT "H-esadecimale"
20 PRINT:PRINT
30 INPUT "NUMERO :";IN$
35 MI$="+"
40 A$=LEFT$(IN$,1):B$=MID$(IN$,2):IF B$=
"" THEN 5000
45 IF ASC(B$)=43 OR ASC(B$)=45 THEN MI$=
B$:B$=MID$(B$,2)
50 IF A$="H" THEN 150
60 IF A$="B" THEN 200
70 IF A$<>"D" THEN 5000
75 DEC=VAL(B$)
80 IF DEC>2^15 THEN PRINT:PRINT "NUMERO
TROPPO GRANDE!":PRINT:GOTO 10
82 IF ASC(MI$)=45 THEN DEC=65536!-DEC

85 PRINT:PRINT "Lo converto in:";:
90 GET IN$:IF IN$="D" OR IN$="H" OR IN$=
"B" THEN PRINT IN$
100 IF IN$="D" THEN V$=STR$(DEC):GOTO 40
00
110 IF IN$="H" THEN 300
120 IF IN$="B" THEN 400
130 GOTO 90
150 REM CONVERSIONE HEX-DEC
155 FOR I=1 TO LEN(B$):M$=MID$(B$,I,1)
160 IF M$<"0" OR M$>"F" THEN 5000
165 NEXT:DEC=0
170 FOR I=1 TO LEN(B$):M$=RIGHT$(B$,I)
175 N=ASC(M$)-48+7*(ASC(M$)>64)
180 DEC=DEC+N*16^(I-1):NEXT
185 IF DEC>2^16 THEN 80
190 GOTO 85
200 REM CONVERSIONE BIN-DEC
210 FOR I=1 TO LEN(B$):M$=MID$(B$,I,1)
220 IF M$<>"0" AND M$<>"1"  THEN 5000
230 NEXT:DEC=0
240 FOR I=1 TO LEN(B$):M$=RIGHT$(B$,I)
250 DEC=DEC-(ASC(M$)=49)*2^(I-1):NEXT
255 IF DEC>2^16 THEN 80
260 GOTO 85
300 REM  CONVERSIONE DEC-HEX
310 V$="":FOR I=3 TO 0 STEP -1
320 N=INT(DEC/(16^I)):DEC=DEC-N*16^I
330 V$=V$+CHR$(48+N-7*(N>9)):NEXT
340 V$=LEFT$(V$,2)+" "+MID$(V$,3):GOTO 4
000
400 REM CONVERSIONE  DEC-BIN
410 V$="":FOR I=15 TO 0 STEP-1
420 N=INT(DEC/(2^I)):DEC=DEC-N*2^I
430 V$=V$+CHR$(48+N):NEXT
440 V$=LEFT$(V$,8)+" "+MID$(V$,9):GOTO 4
000
4000 PRINT:PRINT "IL RISULTATO E':"V$
4010 GOTO 10
5000 PRINT:PRINT "HAI COMMESSO UN ERRORE
 NEL BATTERE IL NUMERO!"
5010 GOTO 10
LISTATO 1. Programma per le conversioni
nei tre sistemi di rappresentazione nume
rica: ESA-DEC-BIN.
```

Listato 1. *Il programma BASIC.*

Anno nuovo, novità JACKSON



Dopo lo strepitoso successo di EI

«la prima e unica Enciclopedia di Elettronica e di Informatica», il Gruppo Editoriale Jackson annuncia che dal 1° febbraio sarà in edicola

ABC Personal Computer:

il Corso di Basic in 24 fascicoli settimanali più facile, più rapido, più completo. Realizzato per voi da chi l'informatica la conosce davvero!





ABC PERSONAL COMPUTER **24 fascicoli settimanali** per ottenere uno splendido Corso rilegato composto da:

2 volumi di **Lezioni** per complessive 608 pagine
1 volume di **Computer-test** di 104 pagine
1 Dizionario di Informatica di 200 pagine

Abbonamento-risparmio + Libro

Tagliando da inviare in busta chiusa a:
Gruppo Editoriale Jackson "ABC Personal Computer"
via Rosellini, 12 - 20124 Milano

Sì, desidero sottoscrivere l'abbonamento risparmio ai 24 fascicoli di **ABC Personal Computer** e alle copertine dei 4 volumi dell'opera. Tutto al **prezzo speciale di L. 80.000** invece di L. 96.000. In più avrò diritto a ricevere immediatamente il volume di Adam Osborne: **Microelettronica, la Nuova Rivoluzione Industriale.**

Allego alla presente
☐ assegno non trasferibile di L. 80.000 a voi intestato
☐ fotocopia di versamento di L. 80.000 sul ccp n. 11666203
☐ fotocopia di vaglia postale di L. 80.000 a voi intestato

I fascicoli dovranno essere inviati a:

Nome ____________ Cognome ____________
Via ____________
Città ____________ Prov. ______ C.A.P. ______
Data ____________ Firma ____________

ABC PERSONAL COMPUTER **il risultato dell'esperienza la conferma della superiorità**

GRUPPO EDITORIALE JACKSON

Introduzione all'intelligenza artificiale

— Parte seconda —

Prosegue in questa seconda parte il nostro "viaggio" nell'intelligenza artificiale

di *Bruno Del Medico*

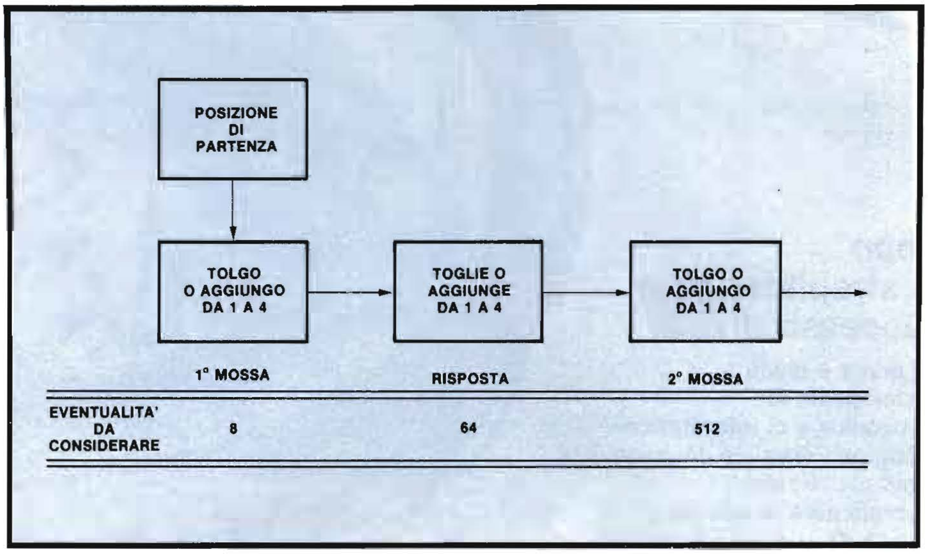


Figura 1. *Il computer non può adottare uno schema logico simile a quello generalmente adottato da un giocatore umano, infatti l'umano può considerare solo le più interessanti tra le mosse possibili, le eventuali risposte dell'avversario, le contromosse.*
Questi tre soli passaggi comportano, nel caso de Il computer impara, la necessità di considerare ben 512 eventualità. Non potendo selezionare quelle "più interessanti" il computer dovrebbe analizzarle tutte e 512 senza ottenere, peraltro, risultati apprezzabili sul piano della strategia.

Nel corso della prima parte, descrivendo lo svolgimento dei programmi presentati, abbiamo fatto alcune considerazioni sulla natura dell'intelligenza artificiale. In particolare abbiamo osservato che un determinato programma si comporta in modo intelligente quando segue schemi logici simili a quelli che seguirebbe un essere umano posto di fronte al medesimo problema.

Era il caso del programma Gara di Master Mind, ed è anche il caso di Tobia nel labirinto, programma presentato in questo numero.
Si nota però che in determinate occasioni il computer segue schemi logici e comportamenti del tutto originali, o comunque abbastanza dissimili da quelli che un essere umano sarebbe tentato di seguire come primo istinto. Questo avviene nel programma Il computer impara.

Il computer impara

Si tratta del programma Almeno una volta, presentato nel numero precedente. Si sono introdotte le modifiche necessarie a rendere più equa la competizione, ed è stato cambiato il nome perché il problema di battere il computer almeno una volta non sussiste più.

Il listato 1 contiene la versione per lo Spectrum, ed il listato 2 contiene le modifiche necessarie per rendere il listato 1 compatibile allo ZX81.

L'impostazione del gioco è rimasta la stessa: il computer visualizza sulla destra dello schermo dei cubetti (questa volta il numero cubetti varia ad ogni partita) ed i giocatori a turno possono toglierne o aggiungerne da 1 a 4. Chi toglie l'ultimo perde. Le aggiunte si possono fare solo se i cubetti sono meno di 31 e più di 10.

Il gioco si basa sulla ricerca di posizioni sicure. Nella versione precedente il computer vinceva sicuramente perché il programma era studiato in modo che il giocatore non potesse uscire dalla sequenza di posizioni perdenti:

26-21-16-11-6-1

Ora, nella nuova versione, la situazione è invertita. Lo sfidante conosce la strategia nei dettagli, e sa che per vincere basta far cadere il computer nella sequenza perdente. Cioè che, togliendo o aggiungendo da 1 a 4 cubetti per volta, occorre farne rimanere sullo schermo solo 21, solo 16 e così via fino a che non ne rimane solo uno. Chi cade una volta nella sequenza non riesce più ad uscirne, se il suo avversario la conosce.

All'inizio del gioco il computer non la conosce: sa solo che può togliere o aggiungere da uno a 4 cubetti per volta. Evidentemente perderà le prime partite, ma imparerà memorizzando tutte le sequenze vincenti giocate, sia da lui che dal gio-

catore.
Dopo poche partite ha in memoria una quantità di sequenze buone sufficienti a metterlo alla pari con lo sfidante.
In pratica avviene che diventa bravo tanto più rapidamente quanto l'avversario è abile.

La tecnica usata dal computer non è immediatamente comprensibile.
Ognuno di noi, affrontando in modo intelligente il gioco, sarebbe portato a ragionare in questo modo:

- posso fare questa mossa;
- il mio avversario potrebbe rispondere con queste altre mosse;
- a seconda della sua risposta, io potrei ...;
- oppure posso fare quest'altra mossa...

Ma questo modo di affrontare il problema è molto dispersivo. Lo schema della figura 1 illustra il totale delle eventualità da considerare, in relazione a tre soli passaggi.
Probabilmente un giocatore umano riuscirebbe a considerare solo le più importanti tra le 512 eventualità possibili, ma il computer non ha la possibilità di discriminare le eventualità più interessanti dalle altre. Dovrebbe inevitabilmente prenderle in considerazione tutte 512, ma come? Forse con 512 linee IF... THEN?

In ogni caso, i risultati di un simile sforzo sarebbero sicuramente inadeguati perché limitati a tre soli spostamenti.
Il computer adotta invece un metodo diverso, illustrato nello schema a blocchi della figura 2.

Analisi del listato 1

La linea 90 inizializza una matrice

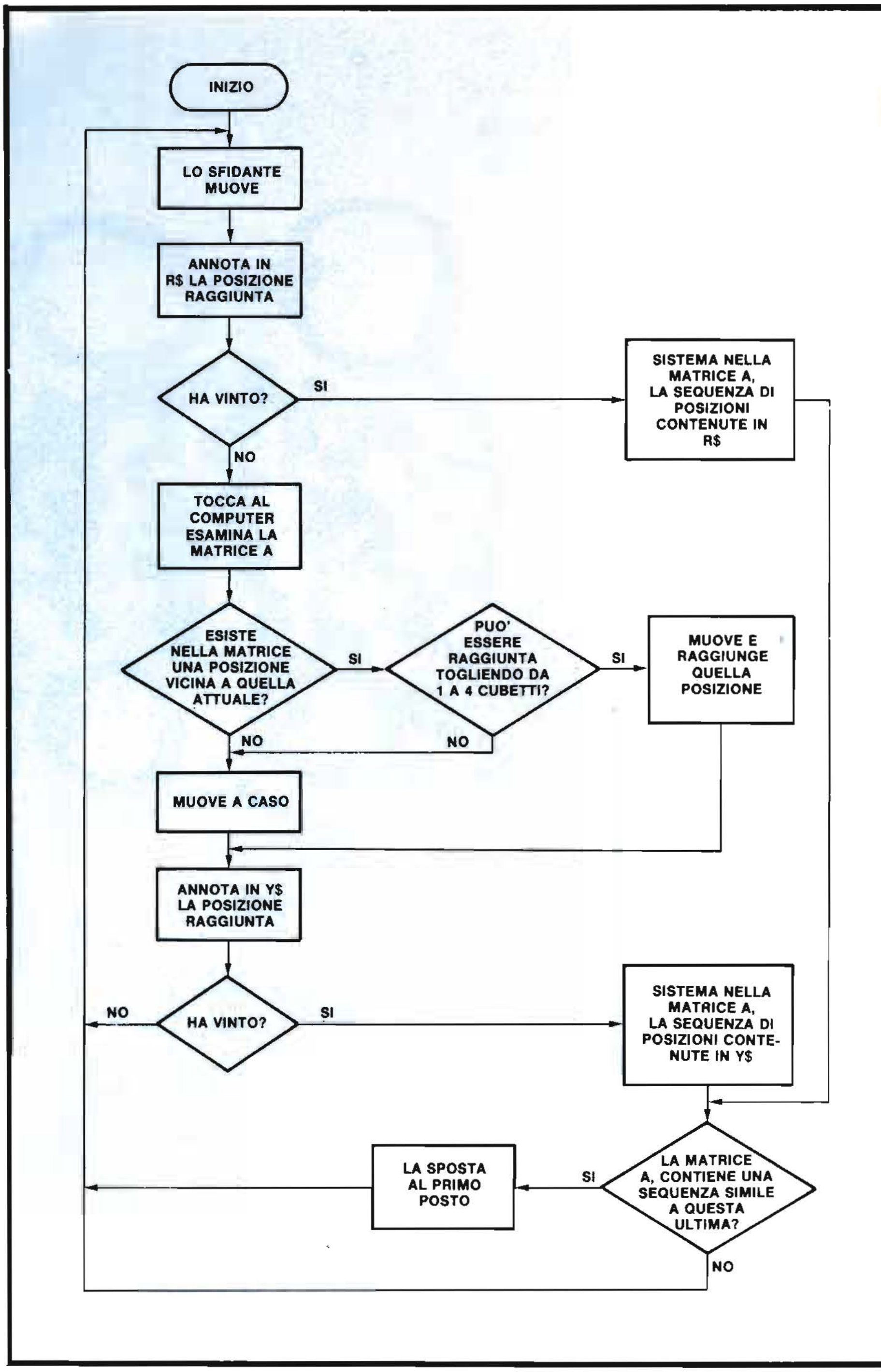


Figura 2. *Lo schema a blocchi in figura illustra il metodo adottato dal computer per migliorare progressivamente la propria strategia di gioco nel programma Il computer impara. In pratica il computer memorizza le sequenze di posizioni giocate dal vincitore di ogni partita.*

E ORA CHE STO ANNEGANDO NELLE CARTE, CHI MI DARA' UNA MANO?



GGK

IL PERSONAL COMPUTER IBM IL TUO PICCOLO GRANDE AMICO.

Un amico che può aiutarti a venire fuori dalla montagna di pratiche che ti sommergono.
Il Personal Computer IBM, così piccolo da stare comodamente sulla tua scrivania, può fare moltissimo per te: aiutarti a risolvere facilmente i problemi quotidiani del tuo lavoro. E non solo quelli. Preventivi, calcoli, contabilità, statistiche, tasse, indirizzi e corrispondenza. Tutto sarà in ordine, perfettamente aggiornato, e stampato in pochissimo tempo.
Non è necessario essere un addetto ai lavori per imparare a usarlo, perchè si fa capire senza difficoltà. Vedrai, in poche ore tu e il tuo Personal Computer IBM diventerete ottimi amici.
Vuoi metterlo alla prova? Vai subito dal tuo concessionario per il Personal Computer IBM. Il tuo tempo è prezioso.

IBM

IBM Italia
Distribuzione Prodotti srl



Il Personal Computer IBM contiene un microprocessore a 16 bit e una memoria di utilizzo che raggiunge i 640 Kbyte, e può essere dotato di un video a colori e di un coprocessore matematico. E, grazie ai dischi fissi, la capacità massima di memoria del sistema è di 21 Mbyte in linea. Inoltre, puoi facilmente collegarti con un altro Personal Computer IBM, con elaboratori più potenti e con la rete dei Centri Servizi Elaborazione Dati della IBM.

Sistemi operativi: DOS 1 - DOS 2 - UCSD - CP/M-86. **Supporti per le comunicazioni:** Asincrone - SDLC - BSC - Emulazione: 3101-3270. **Linguaggi:** tutti i principali e in più l'APL. **Programmi applicativi:** Corso Autodidattico Interattivo - EasyWriter (anche in italiano) - Multiplan (anche in italiano) - VisiCalc - Gestione Aziendale - Contabilità Semplificata

numerica di nome A, composta da 25 vettori: ogni vettore contiene 12 elementi.
Appena inizializzati, tutti gli elementi di tutti i 25 vettori hanno valore zero. Al termine della prima partita, il primo vettore della matrice contiene la sequenza di posizioni occupate da chi ha perso la partita, cioè le posizioni in cui il vincitore ha costretto l'avversario.

Il secondo vettore della matrice A conterrà la sequenza di posizioni giocate dal vincitore della seconda partita. Complessivamente il computer potrà immagazzinare 25 sequenze vincenti.
Quando lo sfidante muove, toglie o aggiunge alcuni cubetti e raggiunge una determinata posizione, quella posizione viene conservata nella variabile stringa R$. (Linee 302 e 672). La posizione che il computer raggiunge muovendo viene conservata nella variabile stringa Y$. (Linee 303 e 2905).

Al termine della partita, la sequenza relativa al vincitore viene sistemata nella stringa T$ (linee 4035 o 4222).
Il ciclo FOR Z (linee 4520-4600) sistema provvisoriamente la sequenza vincente in un vettore di nome B.
Questo vettore viene confrontato con tutti i 25 vettori che compongono la matrice A (linee 4610-4670). Se il computer constata di avere già in memoria una sequenza vincente simile, vuol dire che quella sequenza è migliore delle altre perché si ripete più spesso. La sposta allora nel primo vettore della matrice A, facendo slittare verso il basso tutte le altre.

In questo modo se, per caso, la prima partita è stata vinta in modo fortuito con una sequenza giocata a caso, questa sequenza dopo un certo numero di partite comincia a slittare verso il basso, ed ai primi posti compaiono sequenze molto simili a quella illustrata all'inizio dell'articolo. Se invece il giocatore vince la sua prima partita usando la sequenza vincente, anche il computer sarà in grado di usarla (e la userà efficace-

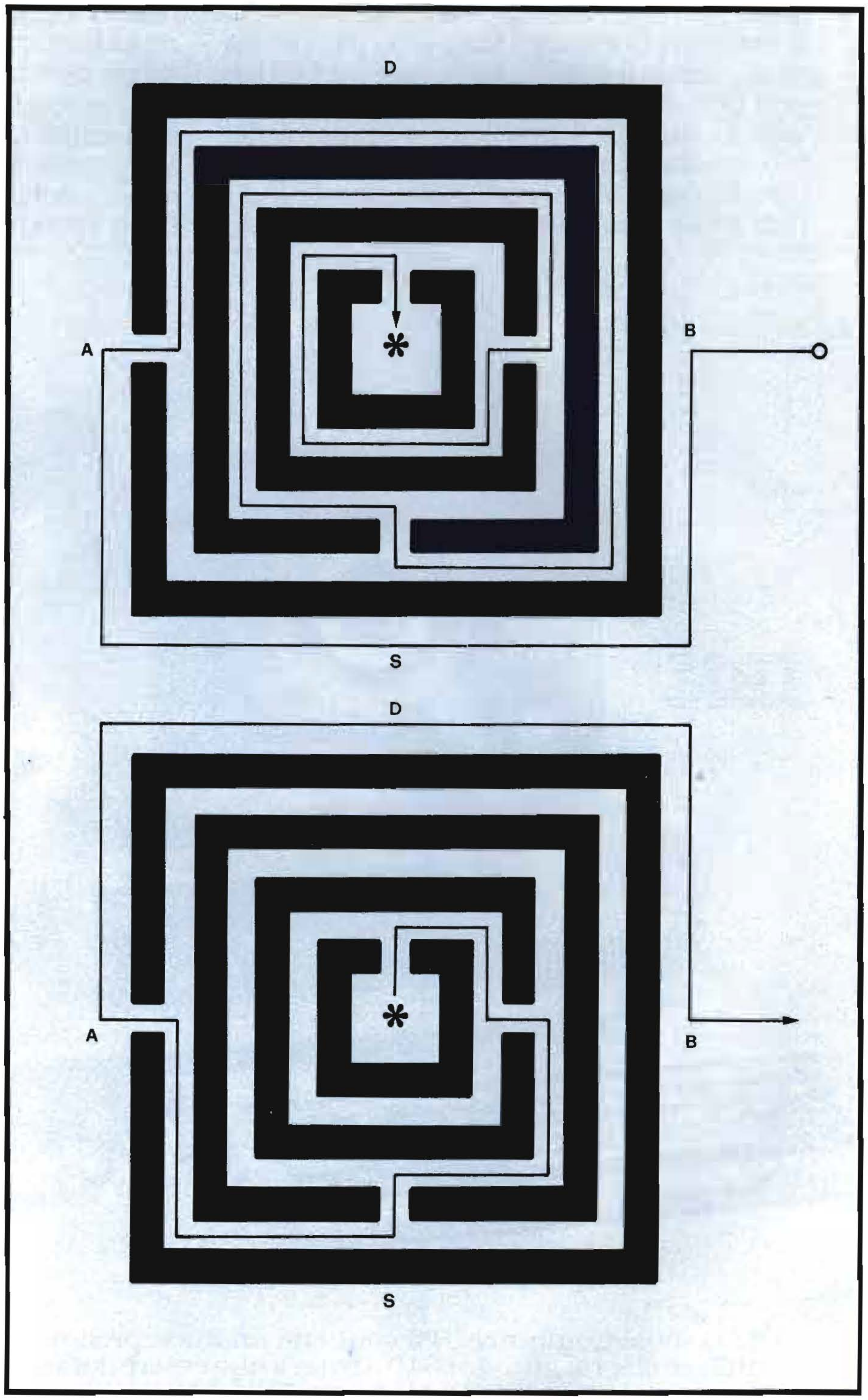


Figura 3. *Quando il bruco Tobia entra per la prima volta nel labirinto, segue sempre la direzione a sinistra per cui può accadere, se il labirinto è uguale a quello illustrato in figura, che la strada sia molto più lunga del necessario (in alto).*
Però percorrendo il labirinto ne studia la composizione ed in particolare le posizioni delle porte. Al ritorno segue senza esitazioni la strada più corta (in basso).

ECCO CHI TI AIUTERÀ AD ANDARE D'AMORE E D'ACCORDO CON IL TUO NUOVO AMICO.



Il tuo concessionario IBM. Ti aiuterà a ottenere il massimo dal tuo Personal Computer IBM. Ti garantirà un'assistenza puntuale e un servizio all'altezza del nome IBM, che in tutto il mondo significa efficienza e affidabilità. Per una lunga e proficua amicizia fra te e il tuo Personal Computer IBM. Per acquisti superiori alle 20 unità puoi anche rivolgerti alle filiali IBM. E per ulteriori informazioni su eventuali punti di vendita che non compaiono sull'elenco, telefona a: 02/21752360 oppure 06/54864962.

ABRUZZI/MOLISE
Pescara - ITALDATA SRL - Via Tiburtina, 75 - Tel. 085.50843
Campobasso - PUBLISISTEMI SRL - Via S. Antonio Abate, 236 - Tel. 0874.98144

BASILICATA
Potenza - I.P.E.S. SPA - Via Sanremo, 79 - Tel. 0971.43293

CALABRIA
Cosenza - CALIÒ SRL - Via N. Serra, 90 - Tel. 0984.32807

CAMPANIA
Cava dei Tirreni - METELLIANA SPA - Via Mandoli, 16 - Tel. 089.463877
Napoli
ENGINEERING INFORMATICA SRL - Via Carducci, 15 - Tel. 081.402660
INFORMATICA MERID. SNC - Via P. Castellino, 179 - Tel. 081.464022
POINTER SISTEMI SRL - Via A. De Gasperi, 45 - Tel. 081.312312
Salerno - OMNIA SRL - C.so Garibaldi, 47 - Tel. 089.220366
S. Maria Capuavetere - GENERAL SYSTEMS SRL - Via Unità d'Italia, 21/23 - Tel. 0823.811100

EMILIA
Bologna
ABACO SAS - Via Bernini, 1 - Tel. 051.393274
CMB INFORMATICA SCRL - Via Arcoveggio, 74/10 - Tel. 051.323594
LUCKY SYSTEMS SRL - Via Farini, 33/A - Tel. 051.231569
SYSDATA ITALIA SPA - Via Massimo d'Azeglio, 58 - Tel. 051.330021
Carpi
DATA SRL - Via B. Peruzzi, 12 - Tel. 059.688090
UNIDATAX SRL - Viale N. Biondo, 6 - Tel. 059.698355
Faenza - DATA SERVICE SRL - Via Laderchi, 2 - Tel. 0546/660300
Ferrara - MARKITALIA COMPUTERS SRL - Via Bologna, 84 - Tel. 0532.96128
Forlì
C.E.DA.F. COOP. ELAB. DATI - Via Zanchini, 57 - Tel. 0543.65402
I.C.O.T. IMPIANTI SRL - Via Codazzi, 10 - Tel. 0543.724246
Imola - PALAZZO DONATO - Via Emilia, 23/A - Tel. 0542.29195
Modena - MASETTI ELETTRONICA SRL - Corso Canalgrande, 14 Tel. 059.219801
Piacenza - RCM COMPUTER SAS - C.so Vittorio Emanuele II, 96 - Tel. 0523.37656
Reggio Emilia
ABAX INFORMATICA SCRL - Via M.K. Gandhi, 1/D - Tel. 0522.26941
A.P.E.D. ELABORAZIONE DATI - Via Filippo Re, 17 - Tel. 0522.38721
MEMAR ELECTRONIC SRL - Via M. Melato, 13 - Tel. 0522.94230
Rimini - HARD & SOFT SYSTEMS SRL - Viale Valturio, 43 - Tel. 0541.773343

LAZIO
Frosinone - SAIU ELETTRONICA SRL - Via Vado del Tufo, 85 - Tel. 0775.83093
Roma
CERVED SPA - Via Appia Nuova, 696 - Tel. 06.7940241
DATAOFFICE SPA - Via Sicilia, 205 - Tel. 06.4754568
ELEDRA 3S SPA - Via G. Valmarana, 63 - Tel. 06.8127324
EXPO SAS - Via 4 Novembre, 151 - Tel. 06.6794293
GEDIN SRL - L.go D. De Dominicis, 7 - Tel. 06.432183
I.S.E.D. SPA - Via Tiburtina, 1236 - Tel. 06.4125851
ISI ITAL SISTEMI PER L'INFORMATICA SPA - P.zza SS Apostoli, 66 - Tel. 06.6793477
ITALSIEL SPA - Via Tevere, 26 - Tel. 06.84311
JACOROSSI SPA - Via V. Brancati, 64 - Tel. 06.50091242
MEMORY COMPUTERS SRL - Via Aureliana, 39 - Tel. 06.4758366
MICROCOMP SPA - V.le M. Gelsomini, 28/30 - Tel. 06.5778484
NICA DIFF INF SRL - V.le Parioli, 40 - Tel. 06.872603
SAPES SRL - V.le Tito Livio, 12 - Tel. 06.3453536
VALDE ADEL SRL - P.zza S. Anastasia, 3 - Tel. 06.6786663
Viterbo - ITALBYTE SRL - V.le Trento - Pal. Garbini - Tel. 0761.221333

LIGURIA
Genova
DIFF.EL. SRL - Via XX Settembre, 31/4 - Tel. 010.586238
ELABORATION PROCESSES SRL - Via Brigata Lig., 68/70/72/74 - Tel. 010.565704
SISTEX SRL - Via SS. Giacomo e Filippo, 13R - Tel. 010.873444
La Spezia - DIFF.EL. CESA SRL - Viale S. Bartolomeo, 139 - Tel. 0187.505223
Sanremo - DIFF.EL. RCS SRL - Via Helsinore, 8 - Tel. 0184.72435

LOMBARDIA
Albino - NUOVA INFORMATICA SAS - Via Provinciale, 86 Comenduno - Tel. 035.751784
Assago - TRANSDATA SRL - Mi Fiori Pal. E3 Str. 1 - Tel. 02.8242460
Bergamo - SELTERING SPA - Via Verdi, 31 - Tel. 035.248256
Brescia
FIN-ECO SERVICE SRL - Via Pastrengo, 5 - Tel. 030.59055
MICROSELT SRL - Via Cipro, 33 - Tel. 030.224246
SELTERING SPA - Via Cipro, 33 - Tel. 030.220391
Bresso - C.I.S.I. SAS - V. Vittorio Veneto, 111 - Tel. 02.6105798
Como - BRUNO SRL - Via Rubini, 5 - Tel. 031.260538
Lecco - ZECCA UFFICIO SPA - Viale Dante, 14 - Tel. 0341.373291
Lodi - ZUCCHETTI SPA - C.so Mazzini, 39 - Tel. 0371.54827
Mantova
REPLICA COMPUTER SRL - Via G. Arrivabene, 22 - Tel. 0376.368821
ANTEK COMPUTER SAS - Via Cavour, 69/71 - Tel. 0376.329333
Milano
AMUFFICIO SAS - Via Desenzano, 7 - Tel. 02.4080275
B.O.M. SAS DI L. LEPRI & C. - V.le Tunisia, 50 - Tel. 02.6598076
COMPUTER SHARING NORD SRL - Piazza S. Maria Beltrade, 1 - Tel. 02.860586
C.S.A. COMM. SRL - Via Farini, 82 - Tel. 02.6888433
CTC GROUP SRL - Via Dante, 14 - Tel. 02.661450
DATA OPTIMATION SRL - Via Masaccio, 12 - Tel. 02.4987876
ECS ITALIA SRL - C.so Monforte, 15 - Tel. 02.780213
EDELEKTRON SRL - C.so Sempione, 39 - Tel. 02.3493603
ELEDRA 3S SPA - Viale Elvezia, 18 - Tel. 02.349751
GENERAL ELECTRIC INFORMATION SERVICES SPA - V.le Regina Giovanna, 29 - Tel. 02.2870181
HOMIC PERSONAL COMPUTER SRL - Piazza De Angeli, 3 - Tel. 02.4988201
HUGNOT LUIGI LUCIANO - Via De Togni, 10 - Tel. 02.873190
IL NUOVO UFFICIO SISTEMI SNC - Via Priv. del Don, 2 - Tel. 02.8350780
ISTITUTO SUPERIORE DI INFORMATICA SRL - Via Montepulciano, 11 - Tel. 02.6701779
MICROTECH SRL - Via F.lli Bronzetti, 20 - Tel. 02.733609
O.E.M. SRL - C.so Sempione, 8 - Tel. 02.3492136
SIRIO SHOP SRL - Viale Certosa, 148 - Tel. 02.3010051
SOFTEC COMPUTER SRL - Viale Maino, 10 - Tel. 02.7491196
S.D.I. STUDIO DI INFORMATICA SPA - Via G. Winckelmann, 1 - Tel. 02.4227361
Monza - EDICONSULT SRL - Via Rosmini, 3 - Tel. 039.389850
Pavia - I.T.C. INFORMATICA SRL - Strada Nuova, 86 - Tel. 0382.303201
Sondrio - G.P.D. DOMENIGHINI SRL - V.le N. Sauro, 28 - Tel. 0342.218561
Varese
ELMEC SPA - Via Sebenico, 12 - Tel. 0332.264135
I.R.P.E. INF. ELETTR. SRL - Via Morazzone, 8 - Tel. 0332.238533
VEGA SPA - Via Silvestro Sanvito, 103 - Tel. 0332.229374
Vigevano - LOGICA INFORMATICA SRL - Via Montegrappa, 32 - Tel. 0381.81888
Vimercate - DATA PROGRES SRL - Via V. Emanuele, 44/A Tel. 039.667423
Vimodrone - OMEGA DATA SRL - Strada Padana Sup., 317 - Tel. 02.2504121

MARCHE
Jesi - SYSTEM HOUSE A.P.R.A. SRL - V.le Cavallotti, 9 - Tel. 0731.58743
Moie - S.E.D.A. SPA - P.zza S. Maria - Tel. 0731.70345
Pesaro - COMPUTER & OFFICE SRL - Via Mazzini, 73 - Tel. 0721.64170

PIEMONTE
Alessandria - INFORMATICA SERVICE SRL - Via Isonzo, 63 - Tel. 0131.445817
Asti - HASTA DATI SNC - Via Silvio Morando, 6/A - Tel. 0141.216356
Biella
TEOREMA SRL - Via Losana, 9 - Tel. 015.24915
V.I.P. COMPUTERS SRL - Via Repubblica, 39 - Tel. 015.27106
Borgosesia - I.D.S. INF. DATA SYST. SRL - Viale Varallo, 157 - Tel. 0163.25327
Cuneo - SISTEMI SRL - Via Giolitti, 26 - Tel. 0171.55475
Genola - EUROSISTEMI SPA - Bivio S.S. 20/28 - Tel. 0172.68176
Novara - ASA SRL - Corso Italia, 25 - Tel. 0321.28250
Torino
DIVERSIFICATE VENCO SRL - C.so Matteotti, 32A - Tel. 011.545525
PROGRAMMA SPA - C.so Svizzera, 185 - Tel. 011.746421
SISTEMI SPA - C.so Peschiera, 249 - Tel. 011.3358676
SOFTEC SRL - C.so San Maurizio, 79 - Tel. 011.8396444
Verbania (Intra) - S 80 SCRL - Via Roma, 7 - Tel. 0323.41083
Vercelli - ANALOG SNC - Via Dionisotti, 18 - Tel. 0161.61105

PUGLIE
Bari
COMPUTER SHARING SUD SPA - Via Trento, 3 - Tel. 080.339177
PASED SRL - Via Calefati, 134/136 - Tel. 080.481488
PUBLISISTEMI SRL - Via Castromediano, 131 - Tel. 080.583943
SIRCOM SRL - Via della Repubblica, 67/69 - Tel. 080.364674
Foggia - MASELLI PER L'UFFICIO - Via L. Zuppetta, 355A - Tel. 0881.78014
Lecce - I.P.E.S. SPA - Via Oberdan, 29 - Tel. 0832.33904
Maglie - S.V.I.C. SRL - Via V. Emanuele, 121 - Tel. 0836.21604

SARDEGNA
Cagliari - C.D.S. SAS - Via Sonnino, 108 - Tel. 070.650756
Olbia - C.P.S. SRL - Via Galvani, 4 - Tel. 0789.51194

SICILIA
Catania
ASIA COMPUTER SRL - Via S. Eupilio, 13 - Tel. 095.326944
COMPUTER SYSTEMS SRL - Via Ruggero di Lauria, 87 - Tel. 095.493777
Messina - SICIL FORNITURE SPA - Via Don Blasco, 75 - Tel. 090.2923987
Palermo
SER.COM. ITALIA SRL - Via Sciuti, 180 - Tel. 091.261041
SI.PR.EL. SRL - Via Serradifalco, 145 - Tel. 091.577344
TESI SRL - Via E. Notarbartolo, 23 - Tel. 091.260549
Trapani - TESI SRL - Via Palmerio Abate, 2 - Tel. 0923.20026

TOSCANA
Empoli - SESA DISTRIBUZIONE SRL - Via XI Febbraio, 24/B - Tel. 0571.72148
Firenze
DATA COOP SCRL - Via di Novoli, 23/H - Tel. 055.4379868
DISTAL SRL - Via Pacini, 46 - Tel. 055.350669
SESA DISTRIBUZIONE SRL - Lungarno Ferrucci, 19R - Tel. 055.6811652
Prato - C.C.S. SAS - Viale Repubblica, 298 - Tel. 0574.580222
Siena - SILOG SRL - Via Sicilia, 5 - Belverde Monteriggioni - Tel. 0577.54085
Viareggio - DELPHI SRL - Via Aurelia Sud, 39 - Tel. 0584.395161

TRE VENEZIE
Bassano D/Grappa - C.P.E. - Piazzetta Poste, 9 - Tel. 0424.20395
Belluno
DE PRÀ SRL - Via I. Caffi, 18 - Tel. 0437.23243
SCP. COMP. SYST. SRL - Via Feltre, 32 - Tel. 0437.70826
Bolzano - BOPAM SAS - Via C. Battisti, 32 - Tel. 0471.30113
Castelfranco Ven. - EDS SRL - Via S. Pio X, 154 - Tel. 0423.490178
Mestre - BOFFELLI F.LLI G & E SNC - Via Bissolati, 19/21/23 - Tel. 041.951247
Padova
CERVED ENGINEERING SPA - C.so Stati Uniti, 14 - Tel. 049.76073
S.I.C. ITALIA SRL - Via Fistomba, 8 - Tel. 049.45555
SYSTEM ROS SAS - P.zza De Gasperi, 14 - Tel. 049.38412
SO.GE.DA. SPA - Via Marsala, 29 - Tel. 049.655385
S. Donà di Piave - COMPUTIME SRL - Piazza Rizzo, 63 - Tel. 0421.2548
Trento
SEDA SAS - Via Sighele, 7/1 - Tel. 0461.984564
SiGe SNC - COMPUTER SHOP - Via Prato, 22 - Tel. 0461.25154
Treviso - INFORMATICA TRE SRL - Viale della Repubblica, 19 - Tel. 0422.65993
Trieste
DITTA MURRI - Via A. Diaz, 24/A - Tel. 040.733253
SELTED SRL - Via Fabio Filzi, 23 - Tel. 040.61381
Udine
D.E.U. SRL - Via Di Prampero, 3/7 - Tel. 0432.204402
D.E.U. SRL - Via Tavagnacco, 89 - Tel. 0432.482086
Verona
PRAGMA SOFTWARE SRL - Via Carmelitani Scalzi, 20 - Tel. 045.596400
SEVER DI G. SERENI - Via Locatelli, 10 - Tel. 045.31331
Vicenza - ALFA DATA SRL - Via Milano, 110 - Tel. 0444.31865

UMBRIA
Perugia
PRISMA INFORMATICA SRL - Via Campo di Marte, 4 N - Tel. 075.71973
PUCCIUFFICIO SNC - Via XX Settembre, 148/C - Tel. 075.72992
Terni - DPS SRL - Via Pacinotti, 6 - Tel. 0744.58247

VALDAOSTA
Aosta - INFORMATIQUE SAS - Centre Commercial l'Amerique, S.S. 26 - Quart d'Aosta - Tel. 0165.765173

Per maggiori informazioni, compila e spedisci questo tagliando al tuo concessionario di zona.

Nome ____________ Cognome ____________
Società ____________ Tel. ____________
Via ____________ N° ____________
Cap ____________ Città ____________

IBM Italia
Distribuzione Prodotti srl

IBM

mente) fin dalla seconda partita.

Quando il computer muove, esamina i vettori della matrice A, che contengono solo sequenze di posizioni vincenti, un elemento per volta.

Il computer deve cercare di raggiungere le posizioni suggerite da questa analisi, partendo dalla posizione in cui si trova.

Ciò è possibile solo se si verificano due condizioni, controllate dalla linea 2040:

- che il numero di cubetti attualmente sullo schermo (NUM) sia superiore alla posizione vincente da raggiungere;
- che la differenza tra NUM e la posizione vincente non sia superiore a 4.

Se tutte e due le condizioni sono vere, il computer calcola la differenza e gioca quella posizione.

Poiché il controllo inizia sempre partendo dal primo vettore di A, è importante che questo contenga la sequenza che vince più frequentemente.

Dopo una ventina di partite la matrice A avrà sicuramente nei suoi primi vettori sequenze con gli ultimi quattro numeri uguali a 16,11, 6, 1. La sequenza contenente anche il 21 sarà tra le prime se non al primo posto. Questo dipende anche da quante volte lo sfidante l'avrà usata.

La tecnica usata dal computer, è un esempio di Processo di apprendimento, e potete verificare come sia semplice ed efficace.

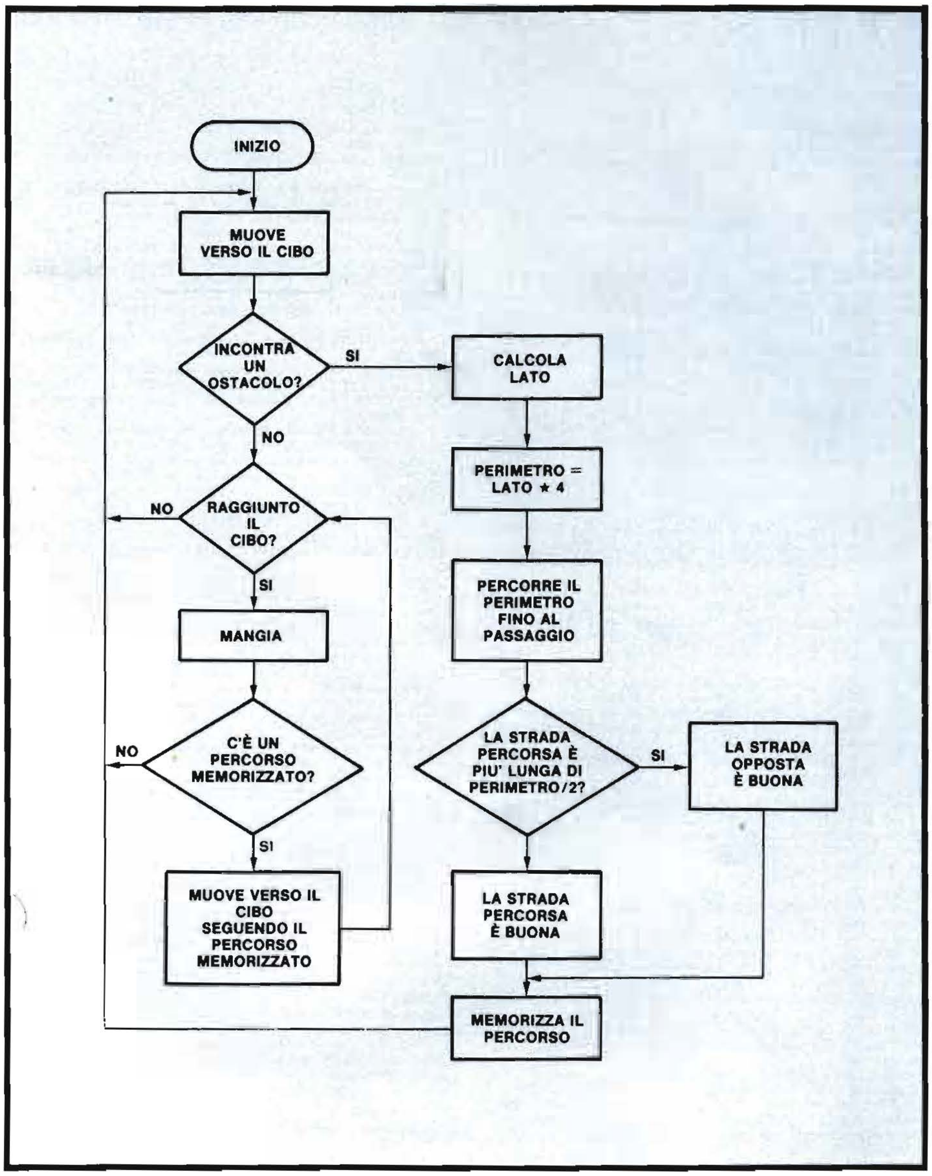


Figura 4. *Comportamento del bruco nel labirinto.*

Tobia nel labirinto (listato 3)

Il bruco Tobia è già noto a chi ci ha seguito nella prima puntata. Questa volta per assicurarsi il cibo deve raggiungere il centro di un labirinto e dopo deve uscirne.

Il labirinto, è generato in modo casuale. All'inizio Tobia non conosce il percorso e procede così: quando incontra un muro gira sempre alla sua sinistra e lo segue fino a quando non trova un passaggio alla sua destra.

Nell'esempio di figura 3 segue il percorso indicato in alto che è molto più lungo del necessario.

Tobia, però, impara e quando esce, segue il percorso più breve che ha ormai memorizzato. Il procedimento seguito è illustrato nello schema a blocchi di figura 4. Sarà bene però analizzare in dettaglio le tecniche di riconoscimento delle strutture del labirinto e di memorizzazione del percorso. Si fa riferimento alla figura 3 ed al listato 3 (per ZX Spectrum).

Svolgimento del programma

Le linee fino alla 205 inizializzano le variabili.

Il ciclo FOR K delle linee 220-500 controlla il movimento del bruco. La linea 250 verifica se la stringa R$ (che, come si vedrà, serve a memorizzare le istruzioni per muoversi nel labirinto) non contiene alcun percorso. Ciò accade quando il labirinto non è mai stato attraversato.

Quando esiste già un percorso, le linee dalla 250 alla 265 si occupano di seguirlo. Diversamente il control-

lo passa alla linea 335 e quindi alle linee 307 e 340 che spostano Tobia, la cui posizione è indicata dalle coordinate (L, C), direttamente verso il cibo, la cui posizione è indicata dalle coordinate (X, Y).

La linea 500 chiude il ciclo e si ritorna alla 220. La linea 235 controlla se il cibo è stato mangiato e in tal caso salta alle linee 600-640 che ristampano il cibo (fuori dal labirinto se prima era dentro). Tutto continua a ripetersi fino a quando Tobia non incontra un muro. La linea 236 rivela quando ciò accade analizzando il colore della posizione di carattere su cui tobia si sta per muovere. Se ATTR (L, C) = 57 significa che la posizione (L, C) è colorata in blu; i muri del labirinto sono blu per distinguerli (si veda il riquadro "l'area degli attributi"). Per lo ZX81 rimane valido quanto detto nella scorsa puntata, in particolare nel riquadro sul display file.

Quando Tobia si trova difronte ad un muro il controllo passa alla parte di programma tra le linee 700 e 900. Le coordinate (L, C) sono quelle della posizione davanti alla testa del bruco. La linea 700 pone la variabile S a 0 se Tobia è su uno dei lati verticali, oppure ad 1 o a 2 se si trova su uno dei lati orizzontali.

Adesso il programma misura il muro, prima dalla posizione di Tobia verso l'alto (o verso sinistra), poi dalla stessa posizione verso il basso (o verso destra). Quando si trova sui lati B o A vengono incrementate le variabili SU e GIU, mentre sugli altri due lati le variabili SIN e DES. La loro somma LATO è la misura del muro in questione, aumentata di uno dato che la casella di partenza è stata contata due volte.

All'inizio Tobia incontra il muro esterno che è lungo 17 caratteri e la variabile LATO conterrà 18. Il passo in più serve perché il bruco segue il muro dall'esterno, e non camminandoci sopra.

Alla linea 790 viene creata per la prima volta la stringa R$ che memorizza il percorso da seguire. All'ini-

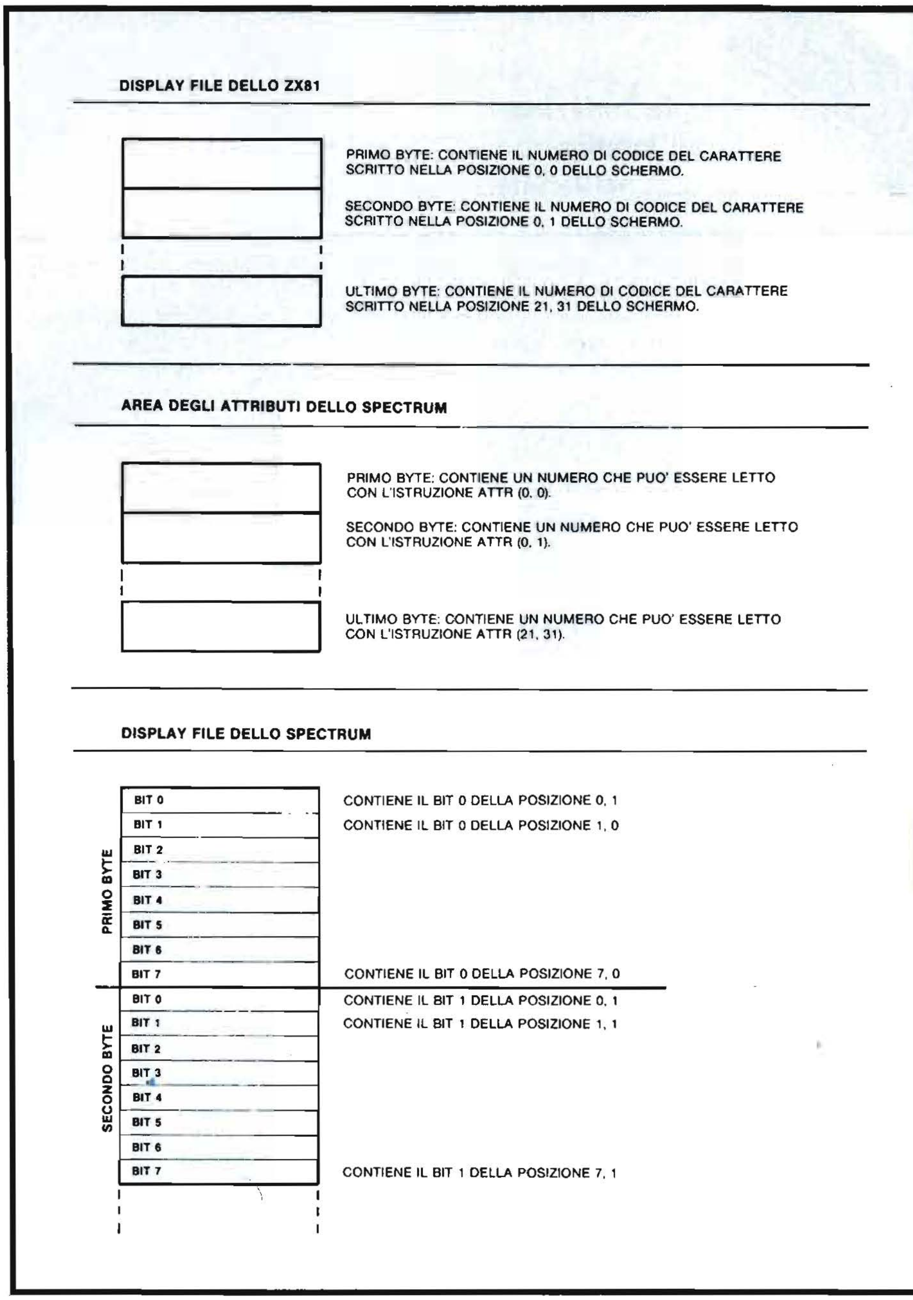


Figura 5. *Confronto tra il Display File dello ZX81, l'area degli attributi ed il Display File dello Spectrum.*

L'area degli attributi nello Spectrum

Nella prima parte di questo articolo abbiamo visto che nello ZX81 ciascuno dei 22 ★ 32 punti indirizzabili sullo schermo con l'istruzione PRINT AT può contenere un carattere in bianco e nero, oppure uno spazio. Il contenuto di ciascuno dei 704 punti può essere ottenuto agevolmente con l'istruzione:

PRINT PEEK 16398 + 256 ★ PEEK 16399

che legge il Display File.

Il contenuto di ciascuno dei 704 punti indirizzabili sullo schermo dello Spectrum, invece, non è sistemato in sequenza nel Display File. Di conseguenza il Display File dello Spectrum, non può essere letto efficacemente, se non ricorrendo ad algoritmi complessi.

Possiamo allora identificare il contenuto di ogni posizione dello schermo dai suoi ATTRIBUTI: questi sono immagazzinati in sequenza nell'area degli attributi, che è composta da 768 byte (comprende anche le due linee più basse).

Nella figura 5 è possibile vedere la differenza tra l'organizzazione della memoria video nello ZX81 e nello Spectrum, e l'organizzazione dell'area degli attributi dello Spectrum.

Facciamo un esempio. Immaginiamo di disegnare sullo schermo una certa quantità di

quadratini neri, diciamo 20, e supponiamo che questi quadratini rappresentano gli alieni che il giocatore deve catturare.

Il giocatore è contrassegnato da una G e si trova nella posizione Y, X per catturare gli alieni deve muoversi fino a soprastamparli.

Una routine che disegni sullo schermo 20 quadratini neri, può essere questa:

```
80 FOR K = 1 TO 20
85 PRINT AT INT (RND ★ 22), INT (RND ★ 32); CHR$ 143
90 NEXT K
```

La linea 85 viene eseguita 20 volte; ogni volta il computer genera due numeri casuali compresi rispettivamente tra 0 e 21 e 0 e 31, e li utilizza come coordinate per disegnare un quadratino nero.

Ma le coordinate in cui si trovano i quadratini sono state generate a caso e non sono conservate in alcun posto della memoria.

Come può il computer accorgersi quando il giocatore va a soprastampare un quadratino nero?

Molte informazioni relative allo schermo vengono conservate dallo Spectrum nel Display File e leggendo questa parte di memoria, dovremmo conoscere il contenuto di ogni parte dello schermo; data la sua particolare configurazione, il Display File non può essere letto con istruzioni PEEK ma solo con la funzione SCREEN$; con l'istruzione:

PRINT SCREEN$ (Y, X)

il computer scrive: G, perché il punto Y, X dello schermo contiene il carattere G (giocatore).

Si può allora ripetere che con l'istruzione:
PRINT SCREEN$ (le coordinate del punto che sta per essere soprastampato dal giocatore)
si ottiene un quadratino nero, quando il giocatore sta per soprastampare un quadratino nero. Ma quanto detto non accade perché la funzione SCREEN$ non riconosce i caratteri grafici e quindi non distingue un quadratino nero da uno spazio vuoto.

A questo punto abbiamo la seguente alternativa:
— usare solo caratteri alfabetici, evitando i caratteri grafici;
— ricorrere ad un altro metodo per riconoscere i caratteri grafici sullo schermo.

Il primo metodo è facilmente realizzabile, possiamo infatti disegnare 20 caratteri A (alieni) al posto dei venti quadratini neri, ma è più conveniente utilizzare i caratteri grafici.

Per conoscerli basta colorarli ed usare l'istruzione ATTR per rilevare il colore di ogni posizione.

Infatti ATTR (W, Z) può avere i seguenti valori:

56 se alle coordinate W, Z è disegnato uno spazio
o un carattere grafico in bianco e nero;
57 se il carattere è colorato in blu;
58 se è colorato in rosso;
59 se è colorato in magenta;
60 se è colorato in verde;
61 se è colorato in ciano;
62 se è colorato in giallo;
63 se è colorato in bianco.

Nella pratica il valore di ATTR si calcola così:

8 moltiplicanto il colore dello sfondo, più il colore del testo

Quando lo sfondo è bianco ATTR vale 8 ★ 7, cioè 56.

Se scriviamo qualcosa in nero il valore di ATTR rimane 56 perché il codice del colore nero è ZERO, per ottenere valori diversi da 56 dobbiamo usare i colori da 1 a 7; cioè dal blu al bianco.

Inoltre il valore di ATTR così ottenuto deve essere aumentato di:
- 64 se la posizione è EXTRA-LUMINOSA, cioè se è sotto l'influenza del comando BRIGHT;
- 128 se la posizione è lampeggiante, cioè se è sotto l'influenza del comando FLASH.

Nel listato 3, Tobia riconosce i muri del labirinto perché sono colorati in blu e il verde di ATTR diventa 57 (linea 236 del listato 3).

zio sarà composta da 18 caratteri "B", più 18 "S", più 18 "A", più 18 "D".

Occorre ora spiegare che in questa stringa, ed in altre analoghe usate nel programma, ogni carattere corrisponde ad un passo del bruco in una certa direzione:
B corrisponde ad un passo verso il basso;
S corrisponde ad un passo verso sinistra;
A corrisponde ad un passo verso l'alto;
D corrisponde ad un passo verso destra.

Alla fine la stringa E$ conterrà in questo modo il percorso più breve da seguire per arrivare al cibo.

La stringa R$ contiene ora le istruzioni per eseguire un giro completo a partire dall'angolo in alto a destra. Tobia però non occupa questa posizione ed ecco che le linee 800-803 manipolano la stringa in modo che le istruzioni partano dalla posizione occupata.

Tobia segue ora le indicazioni della stringa R$ fino a quando le linee 832-835 non rivelano un passaggio verso i corridoi più interni; allora il ciclo FOR M viene abbandonato ed il controllo passa alla linea 850.

La variabile P contiene il numero di passi fatti da Tobia dal momento in cui ha incontrato l'ostacolo fino al raggiungimento del passaggio. Nel caso del nostro esempio, 36 passi: 9 in basso, 18 a sinistra e 9 in alto. Quando P è minore di LATO ★ 2, vuol dire che il passaggio è stato raggiunto con un numero di passi inferiore a quello che sarebbe stato necessario seguendo la direzione opposta.

In questo caso la strada percorsa è effettivamente la più breve e Tobia memorizza questo spezzone di percorso aggiungendolo a quanto già contenuto nella variabile E$. (Linea 900).

Negli altri casi, quando P è maggiore o uguale a LATO ★ 2, Tobia considera buono il percorso opposto cioè quello contenuto nella parte

di stringa R$ non utilizzata. Vediamo in pratica come varia la stringa R$:
a) stringa R$ generata subito dopo il calcolo del lato:
"BBBBBBBBBBBBBBBB ;S
SSSSSSSSSSSSSSSSS
AAAAAAAAAAAAAAAAAA
DDDDDDDDDDDDDDDDDD"

b) stringa R$ modificata in funzione della posizione di Tobia:
"BBBBBBBBB
SSSSSSSSSSSSSSSSSSSSSS
AAAAAAAAAAAAAAAAAA
DDDDDDDDDDDDDDDDDD
BBBBBBBBB"

c) parte di stringa R$ utilizzata da Tobia per seguire il perimetro del labirinto fino al raggiungimento del passaggio (strada non buona):
"BBBBBBBBB
SSSSSSSSSSSSSSSSS
AAAAAAAAA"
d) parte di stringa R$ non utilizzata da Tobia, valutata come percorso buono:
"AAAAAAAAA
DDDDDDDDDDDDDDDDDD"
BBBBBBBBB"
Questa ultima parte di stringa rappresenta il percorso da memorizzare, ma non può essere memorizzata così com'è. Infatti per seguire il percorso inverso, Tobia deve leggere la stringa R$ partendo dall'ultimo carattere anziché dal primo; e dovendosi muovere in senso opposto, deve interpretare "alla rovescia" le indicazioni della stringa. Prima di memorizzare la stringa occorre quindi:
• ribaltare la stringa, in modo che il primo carattere diventi l'ultimo;
• invertire il significato dei comandi: S diventa D, A diventa B, e viceversa.

Queste operazioni vengono eseguite alle linee 870 e 875.
Arrivato di fronte al passaggio e memorizzato il percorso più conveniente, Tobia constata che non esistono ostacoli immediati nel percorso verso il cibo e passa il controllo alle linee 377 e 340, che lo fanno muovere finché non incontra un nuovo ostacolo; cioè dopo due passi quando incontra il muro del perimetro più interno.

Allora il controllo passa ancora alle linee 700 e seguenti. Tobia ripete tutte le operazioni illustrate: calcola la lunghezza del lato ed il perimetro, e segue il perimetro fino ad incontrare il passaggio. Qui valuta nuovamente se sia più breve la strada percorsa o quella opposta e memorizza il percorso conveniente aggiungendolo a quello già contenuto nella stringa E$. Passa poi al secondo corridoio interno e poi al terzo, fino ad incontrare il cibo.

Poco tempo dopo appare del nuovo cibo al di fuori del labirinto, ed il controllo ritorna al ciclo FOR K (linee 220/500). Questa volta però la lunghezza della stringa R$ non è uguale a zero, perché R$ (R$ = E$) contiene il percorso memorizzato quando Tobia ha percorso la prima volta il labirinto.

Ecco quindi che il controllo passa alle linee 260 e 261 e rimane a queste linee finché la stringa R$ non è esaurita cioè finché non è stata ripercorsa tutta la strada (questa volta la più breve) per uscire dal labirinto.
Tobia continua a correre dentro e fuori dal labirinto perché il cibo appare una volta dentro e una volta fuori. Ogni volta la subroutine 2000 effettua il doppio ribaltamento della stringa E$, contenente il percorso completo.
Fermando il programma con il BREAK e dando nuovamente il RUN, la stringa E$ viene azzerata; il computer genera un nuovo labirinto e tutto ricomincia da capo.

Le tecniche descritte sono valide, con gli opportuni adattamenti, anche nel caso di labirinti più complessi.
La subroutine 9250 fa sì che tutto il corpo del bruco si muova in accordo con la sua testa, secondo le modalità descritte nella prima parte dell'articolo.
La variabile T (linea 610 e seguenti) determina la posizione in cui deve essere posto il cibo; se T è uguale a 1 nel labirinto, se T è uguale a 2 fuori dal labirinto. Ne consegue che la variabile T deve assumere per un numero indefinito di volte, alternativamente, i valori di 1 e 2 (linea 610).

Il listato 4 contiene le modifiche necessarie per adattare il listato 3 allo ZX81.

Il gioco Tiro mancino (listato 5) si svolge su una scacchiera con 9 posizioni: le tre in basso sono occupate dalle tre pedine del giocatore (G), le tre centrali sono disponibili per gli spostamenti, e le tre in alto sono occupate dalle pedine del computer (S, per Sinclair).

Il giocatore può muovere solo verso l'alto (Sinclair muove verso il basso) verticalmente, e può mangiare in senso obliquo. Perde chi si trova nella condizione di non poter più muovere.

Per fare la mossa il giocatore batte due numeri: il primo indica la posizione di partenza, il secondo la posizione che vuole raggiungere, sulla sinistra appare una seconda scacchiera, che serve unicamente a ricordare al giocatore i numeri associati alle varie posizioni.

Con il susseguirsi delle partite il computer diventa progressivamente più abile.

Per migliorare la propria strategia memorizza nel vettore A tutte le mosse fatte, in relazione alle varie situazioni di gioco e, quando deve muovere, esamina tale vettore A per vedere se la situazione attuale si è già verificata in precedenza.

Se trova la situazione nel vettore, sceglie a caso una delle mosse consentite rispetto a quella situazione, se invece la situazione è nuova, la sistema nel vettore con tutte le possibili mosse che possono essere fatte partendo da quella situazione.

Se il computer perde la partita, cancella le mosse che lo hanno portato a perdere dalla lista delle possibilità. In questo modo, dopo alcune partite, il vettore P contiene quasi solo indicazioni relative a mosse vincenti.

Introduzione all'intelligenza artificiale

Listato 1. *Questo programma è una nuova versione di "ALMENO UNA VOLTA", presentato nel numero precedente. Ciascuno dei due giocatori può conseguire facilmente la vittoria se conosce la strategia delle posizioni vincenti. All'inizio il computer non conosce questa strategia, ma l'apprende nel giro di poche partite.*
Per ottenere i caratteri speciali delle linee 7116 e 9932 occorre caricare per prime le linee 9900 e 9920 e dare il RUN. Fatto ciò selezionate GRAPHICS e otterrete i caratteri speciali premendo A e B. Caricate normalmente il resto del programma. La versione presentata è relativa allo Spectrum.

```
   1 REM LO SPECTRUM IMPARA
  90 DIM a(25,12)
 100 GO SUB 9900
 112 LET vi=0
 114 LET vt=0
 116 LET cp=1
 118 LET bmd=0
 122 LET ag=0
 124 LET sc=0
 200 GO SUB 9000
 210 PRINT AT 10,0;"COME TI CHIA
MI ?"
 215 INPUT g$
 220 IF LEN g$>10 THEN LET g$=g$
( TO 10)
 225 PRINT g$
 230 PRINT
 240 PRINT "VUOI VEDERE LE REGOL
E ? (S/N)"
 245 PAUSE 20000
 250 IF CODE INKEY$=115 OR CODE
INKEY$=83 THEN GO SUB 8000
 300 LET cm=0
 302 LET r$=","
 303 LET y$=","
 305 LET tg=0
 310 LET jk=0
 320 LET num=INT (RND*12)+15
 360 GO SUB 7000
 500 LET cm=cm+1
 501 LET tg=0
 502 GO SUB 7700
 504 PRINT AT 2,9;cm;AT 3,12;vt;
AT 16,15;"  ";AT 16,15;num
 508 IF cm=1 THEN PRINT AT 18,15
;"TOCCA A TE"
 510 PRINT AT 19,15;"1 PER AGGIU
NGERE";AT 20,15;"2 PER TOGLIERE"
 520 IF INKEY$="" THEN GO SUB 95
00
 530 LET x$= INKEY$
 535 IF CODE x$<>49 AND CODE x$<
>50 THEN GO TO 560
 550 LET sc=VAL x$
 555 IF sc=1 AND num<=10 OR sc=1
 AND num>30 THEN GO TO 560
 556 GO TO 600
 560 LET bmd=510
 562 GO TO 8200
 600 GO SUB 7500
 610 PRINT AT 18,16;"QUANTI NE";
AT 19,16;("TOGLI ?" AND sc=2)+("
AGGIUNGI ?" AND sc=1)
 620 IF INKEY$="" THEN GO SUB 95
00
 630 LET h$=INKEY$
 640 IF CODE h$<49 OR CODE h$>52
 THEN GO TO 680
 650 LET mn=VAL h$
 660 IF sc=2 THEN LET mn=-mn
 670 LET num=num+mn
 672 LET R$=R$+STR$ NUM+","
 675 IF num<1 OR num>31 THEN GO
TO 680
 677 GO TO 700
 680 LET bmd=620
 682 GO TO 8200
 700 GO SUB 1000
 702 GO SUB 7500
 705 PRINT AT 18,16;"NE HAI";AT
19,16;("TOLTI " AND sc=2)+("AGGI
UNTI " AND sc=1)+h$
 706 IF num=1 THEN GO TO 4000
 707 PRINT AT 16,15;"  ";AT 16,1
5;num
 710 GO SUB 7600
 720 PRINT AT 18,1;"TOCCA A ME";
AT 19,1; INVERSE 1;"VADO"
 740 GO SUB 2000
 750 GO SUB 7600
 755 GO SUB 7500
 760 PRINT AT 18,15;("TOLTI " AN
D tg<>0)+("AGGIUN. " AND ag<>0)+
"IO:";tg+ag; INVERSE 1;"VAI:"
 765 IF num=1 THEN GO TO 4200
 770 GO TO 500
 999 STOP
1000 REM stampa la scacchiera
1010 LET l=INT (num/6)*2+6
1020 LET c=16+(num-((INT (num/6)
)*6))*2
1030 IF jk=1 AND sc=2 THEN GO TO
 1500
1040 IF tg<>0 THEN GO TO 1500
1050 FOR k=l TO 6 STEP -2
1060 FOR w=c TO 16 STEP -2
1070 PRINT AT k,w;"■"
1090 NEXT w
1095 LET c=28
1100 NEXT k
1110 LET jk=1
1120 RETURN
1500 LET kj=18
1510 FOR k=l+2 TO l STEP -2
1520 FOR w=28 TO kj STEP -2
1530 PRINT AT k,w;" "
1540 NEXT w
1550 LET kj=c+2
1560 NEXT k
1580 GO TO 1120
1999 REM IL COMPUTER SCEGLIE LA
MOSSA
2000 LET tg=0
2010 LET ag=0
2012 LET sc=0
2020 FOR z=1 TO 25
2025 IF A(Z,12)=0 THEN GO TO 206
0
2030 FOR w=1 TO 12
2032 IF A(Z,W)=0 THEN GO TO 2050
2040 IF num>a(z,w) AND num-a(z,w
)<5 THEN GO TO 2200
2050 NEXT w
2060 NEXT z
2070 LET tg=INT (RND*4)+1
2072 IF num-tg<1 THEN GO TO 2070
2080 GO TO 2900
2200 LET tg=num-a(z,w)
2900 LET num=num-tg
2905 LET y$=y$+STR$ num+","
2910 GO SUB 1000
2920 RETURN
4000 PRINT AT 10,18;"O.K. HAI";A
T 11,18;"VINTO TU"
4010 LET vt=vt+1
4020 PRINT AT 18,15;"
"
4030 PAUSE 20000
4035 LET t$=r$
4040 GO TO 4225
4200 PRINT AT 10,18;"HO VINTO IO
"
4220 LET vi=vi+1
4222 LET t$=y$
```



Personal e home computer

Provando e riprovando

Nicole Bréaud-Pouliquen
La pratica dell'APPLE

Per imparare a usare un calcolatore bisogna... usarlo.
Solo così, ad esempio, è possibile scoprire e sfruttare le immense risorse operative offerte dall'**APPLE**. Provando, riprovando e... leggendo un manuale come questo.
Scritto da un vero esperto, il libro si compone di 3 capitoli fondamentali:
- **Il sistema APPLE II"** dedicato all'hardware e al software
- **"Il BASIC APPLESOFT"** con le istruzioni, i sottoprogrammi, gli operatori aritmetici e logici
- **"Il disegno e la grafica"** con le zone di memoria RAM e le funzioni grafiche.

Il tutto arricchito da numerosi esempi ed esercitazioni con soluzioni: affinchè la pratica abbia l'immediata soddisfazione del riscontro.
130 pagine
Lire 10.000
Codice **341D**



SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	341D	**L. 10.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________
n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato
☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

nuovidea

Seguito listato 1.

```
4225 PRINT AT 19,15; INVERSE 1;"
PREMI UN TASTO";AT 20,15;"PER CO
NTINUARE"
4226 LET cp=cp+1
4227 PRINT AT 16,15; FLASH 1;" 1
"
4228 IF INKEY$="" THEN GO SUB 95
00
4235 GO SUB 4500
4240 GO TO 300
4500 DIM b(12)
4520 FOR z=12 TO 1 STEP -1
4530 LET t$=t$( TO LEN t$-1)
4540 FOR w=LEN t$ TO 1 STEP -1
4550 IF t$(w)="," THEN GO TO 457
0
4560 NEXT w
4570 LET b(z)=VAL t$(w+1 TO )
4580 LET t$=t$( TO w)
4590 IF LEN t$=1 THEN GO TO 4610
4600 NEXT z
4610 FOR z=1 TO 25
4620 LET sw=0
4630 FOR w=12 TO 1 STEP -1
4640 IF a(z,w)=b(w) THEN LET sw=
sw+1
4641 IF sw>=4 THEN GO TO 4800
4645 IF a(z,w)<>b(w) THEN GO TO
4670
4650 NEXT w
4670 NEXT z
4680 FOR z=1 TO 25
4690 IF a(z,12)=0 THEN GO TO 472
0
4700 NEXT z
4710 RETURN
4720 FOR w=12 TO 1 STEP -1
4730 LET a(z,w)=b(w)
4740 NEXT w
4750 RETURN
4800 IF a(25,12)<>0 THEN RETURN
4810 FOR w=1 TO 25
4820 IF a(w,12)=0 THEN GO TO 484
0
4830 NEXT w
4840 FOR j=w TO 2 STEP -1
4850 FOR s=1 TO 12
4860 LET a(j,s)=a(j-1,s)
4870 NEXT s
4880 NEXT j
4890 FOR j=12 TO 1 STEP -1
4900 LET a(1,j)=b(j)
4910 NEXT j
4990 RETURN
7000 REM STAMPA DEL QUADRO
7005 CLS
7010 PRINT "[redacted]"
7015 PRINT AT 21,0;"[redacted]"
7020 FOR k=1 TO 20
7022 PRINT AT k,0;"█";AT K,31;"█
"
7023 PRINT AT k,14;"█"
7025 NEXT k
7030 PRINT AT 4,1;"[redacted]"
7035 PRINT AT 17,1;"[redacted]"
7040 FOR k=4 TO 17
7045 PRINT AT k,14;"█"
7050 NEXT k
7075 PRINT AT 1,2;g$;AT 2,2;"MOS
SA";AT 3,2;"VITTORIE"
7100 PRINT AT 1,18;"LO";AT 2,18;
"SPECTRUM";AT 3,18;"IMPARA"
7110 FOR k=4 TO 8
7111 FOR w=9 TO 12
7112 INK 4: PRINT AT w,k;"█"
7113 NEXT w
7114 NEXT k
```

Seguito listato 1.

```
7116 PRINT AT 10,2;"█";AT 11,2;"
██";AT 13,3;"█";AT 13,6;"█";AT 1
2,9;"█";AT 11,9;"██";AT 10,9;"█"
7120 PRINT AT 10,13;"█";AT 11,12
;"██";AT 12,13;"█";AT 13,12;"██"
7130 INK 0
7140 PRINT AT 13,5;"██";AT 14,6;
"█ ";AT 15,6;" █";AT 16,6;"█"
7145 PRINT AT 14,13;"█ ";AT 15,12
;" █";AT 16,13;"█ "
7199 GO SUB 1000
7200 RETURN
7499 REM 16 spazi
7500 PRINT AT 18,15;"
"
7505 PRINT AT 19,15;"
"
7508 PRINT AT 20,15;"
"
7510 RETURN
7599 REM 13 spazi
7600 PRINT AT 18,1;"
"
7605 PRINT AT 19,1;"
"
7607 PRINT AT 20,1;"
"
7610 RETURN
7700 PRINT AT 18,2;"SPECTRUM";AT
 19,2;"MOSSA ";CM;AT 20,2;"VITTO
RIE ";VI
7710 RETURN
8000 GO SUB 9000
8020 PRINT "A TURNO POSSIAMO TOG
LIERE O AG- GIUNGERE DA 1 A 4 CU
BETTI ALLA  TABELLA CHE VEDRAI S
ULLA DESTRA DEL VIDEO"
8030 PRINT
8040 PRINT "LE AGGIUNTE SI POSSO
NO FARE SOLOQUANDO I CUBETTI SON
O PIU DI 10 O MENO DI 31."
8050 PRINT
8060 PRINT "CHI TOGLIE L'ULTIMO
PERDE.  IL TOCCO SUI TASTI DEVE
ESSERE RAPIDO"
8070 PRINT AT 21,0; INVERSE 1;"P
REMI UN TASTO"
8100 PAUSE 20000
8110 RETURN
8200 LET dmb=bmd
8205 LET bmd=0
8220 PRINT #1;AT 1,0; INVERSE 1;
"INPUT ERRATO  RIPETI PREGO"
8230 PAUSE 120
8235 PRINT #1;AT 1,0;"
"
8250 GO TO dmb
9000 CLS
9020 PRINT TAB 7;"LO SPECTRUM IM
PARA"
9021 PRINT
9030 PRINT "[redacted]"
9040 RETURN
9500 IF INKEY$<>"" THEN RETURN
9555 LET o$=o$(13)+o$: LET o$=o$
( TO 13): PRINT AT 6,1;o$
9556 IF INKEY$<>"" THEN RETURN
9557 LET p$=p$(13)+p$: LET p$=p$
( TO 13): PRINT AT 7,1;p$
9558 IF INKEY$<>"" THEN RETURN
9559 LET q$=q$(13)+q$: LET q$=q$
( TO 13): PRINT AT 8,1;q$
9570 GO TO 9500
9900 DATA "a",85,85,85,85,85,85,
85,85,"b",0,12,2,1,2,12,48,0
9920 FOR z=1 TO 2: READ z$: FOR
w=0 TO 7: READ e: POKE USR z$+w,
e: NEXT w: NEXT z
9932 LET o$="     >      >     "
9934 LET p$=" >         >>>>  "
9936 LET q$=" >         >>    "
9950 RETURN
```

Listato 2. *Modifiche necessarie per rendere il listato 1 compatibile allo ZX81. La linea REM indica quali linee del listato 1 non devono essere considerate, le altre linee si aggiungono al listato se hanno numeri diversi, oppure sostituiscono quelle con numeri uguali. Per lo ZX80 nuova ROM le linee 520 e 620 devono diventare: PAUSE 20000 e si perde l'effetto degli uccelli in volo, a causa dello SLOW non attivato.*

```
   1 REM LO ZX81 IMPARA
   3 REM MODIFICHE AL LISTATO 1
   4 REM CANCELLARE LE LINEE DAL
LA 7113 ALLA 7145,LA 9900 E LA 9
920
 250 IF CODE INKEY$=56 THEN GOSU
B 8000
 535 IF CODE X$<>29 AND CODE X$<
>30 THEN GOTO 560
 640 IF CODE H$<29 OR CODE H$>32
 THEN GOTO 680
 720 PRINT AT 18,1;"TOCCA A ME";
TAB 1;"[illegible]"
 760 PRINT AT 16,15;("TOLTI " AN
D TG<>0)+("AGGIUN." AND AG<>0)+"
IO:";TG+AG;"[illegible]"
4000 PRINT AT 10,16;"[illegible]";AT 11
,18;"[illegible]"
4200 PRINT AT 10,18;"[illegible]
"
4225 PRINT AT 19,15;"[illegible]
[illegible]";AT 20,15;"[illegible]"
4227 PRINT AT 16,15;"[illegible]"
7110 PRINT AT 9,4;"[illegible]";TAB 2;
"[illegible]";TAB 2;"[illegible]";TAB
 4;"[illegible]";TAB 3;"[illegible]"
7111 PRINT AT 14,6;"[illegible]";TAB 6;"[illegible]"
;TAB 6;"[illegible]"
7112 PRINT AT 10,13;"[illegible]";TAB 12;"
[illegible]";TAB 11;"[illegible]";TAB 12;"[illegible]";TAB
 13;"[illegible]";TAB 12;"[illegible]";TAB 13;"[illegible]"
7700 PRINT AT 16,2;"ZX81";TAB 2;
"MOSSA ";CM;TAB 2;"VITTORIE ";VI
```

```
8070 PRINT AT 21,0;"[illegible]
[illegible]"
8210 POKE 16418,0
8220 PRINT AT 23,0;"[illegible]
[illegible]"
8230 PAUSE 120
8240 PRINT AT 23,0;"
"
8250 POKE 16418,2
9500 IF INKEY$<>"" THEN RETURN
9553 LET O$=O$(LEN O$)+O$
9554 IF INKEY$<>"" THEN RETURN
9555 LET O$=O$( TO LEN O$-1)
9556 IF INKEY$<>"" THEN RETURN
9557 PRINT AT 6,1;O$
9558 IF INKEY$<>"" THEN RETURN
9559 LET P$=P$(LEN P$)+P$
9560 IF INKEY$<>"" THEN RETURN
9562 LET P$=P$( TO LEN P$-1)
9563 IF INKEY$<>"" THEN RETURN
9564 PRINT AT 7,1;P$
9566 IF INKEY$<>"" THEN RETURN
9567 LET Q$=Q$(LEN Q$)+Q$
9568 IF INKEY$<>"" THEN RETURN
9569 LET Q$=Q$( TO LEN Q$-1)
9570 IF INKEY$<>"" THEN RETURN
9572 PRINT AT 8,1;Q$
9574 IF INKEY$<>"" THEN RETURN
9580 GOTO 9500
9932 LET O$="     >      >    "
9934 LET P$=" >         >>>> "
9936 LET Q$=" >         >>   "
9950 RETURN
```

Listato 3. *Questa volta il bruco Tobia deve raggiungere il cibo all'interno di un labirinto e deve poi uscire alla svelta per afferrare il cibo che appare fuori. Per fare ciò, la prima volta che entra nel labirinto segue un percorso fisso scegliendo sempre la direzione a sinistra (vedi la figura 3 in alto). Al ritorno però sceglie senza indugi la strada più corta (figura 3 in basso). Entrando nel labirinto, il bruco ne studia la struttura e le posizioni delle porte e utilizza questi dati ogni volta che lo ripercorre. La versione presentata è relativa allo Spectrum.*

```
   1 REM TOBIA NEL LABIRINTO
  70 GO SUB 3000
[illegible]
"": LET SWITCH=0
 100 DIM P(6)
 102 DIM Q(6)
 110 LET X=10
 120 LET Y=10
 130 LET L=10
 140 LET C=25
 150 LET A$=CHR$ 144+CHR$ 145+CH
R$ 145+CHR$ 145+CHR$ 145+CHR$ 32
 160 FOR K=1 TO 6
 170 LET P(K)=L
 180 LET Q(K)=25+K
 190 NEXT K
 195 PRINT AT L,C;A$
 200 GO SUB 9000
 205 PRINT AT 10,10;"*"
 220 FOR K=1 TO 500
 230 PRINT AT L,C;
 235 IF L=Y AND C=X THEN GO SUB
1000: GO TO 600
 238 IF ATTR (L,C)=57 THEN GO TO
 700
 245 IF SWITCH=0 THEN LET R$=E$:
 LET SWITCH=1
 250 IF LEN R$=0 THEN GO TO 335
 260 LET L=L+(R$(1)="B")-(R$(1)=
"A")
 261 LET C=C+(R$(1)="D")-(R$(1)=
"S")
 262 GO SUB 9250
 264 LET R$=R$(2 TO )
```

```
 265 GO TO 250
 335 GO SUB 9250
 336 LET C1=C: LET L1=L
 337 LET C=C-(X<C)+(X>C)
 340 LET L=L-(Y<L)+(Y>L)
 350 GO SUB 1000
 500 NEXT K
 600 PRINT AT Y-1,X; FLASH 1;"AM
"
 605 PAUSE 200
 606 PRINT AT Y-1,X;"  ";AT 10,1
0;" "
 610 LET T=T+1: IF T>2 THEN LET
T=1
 612 IF T=2 THEN GO TO 618
 614 LET Y=10: LET X=26: PRINT A
T Y,X;"*"
 615 GO SUB 2000
 616 LET SWITCH=0: LET R$="": GO
 TO 220
 618 LET Y=10: LET X=10: PRINT A
T Y,X;"*"
 635 GO SUB 2000
 640 LET SWITCH=0: LET R$="": GO
 TO 220
 700 LET S=0+(1 AND ATTR (L,C-1)
=57)+(1 AND ATTR (L,C+1)=57)
 705 LET SU=0: LET GIU=0: LET DE
S=0: LET SIN=0
 715 FOR Z=(L AND S=0)+(C AND S<
>0) TO 0 STEP -1
 720 IF S=0 AND ATTR (Z,C)=56 TH
EN GO TO 750
 721 IF S=0 THEN LET SU=SU+1
```

Introduzione all'intelligenza artificiale

Seguito listato 3.

```
723 IF S<>0 AND ATTR (L,Z)=56 THEN GO TO 750
724 IF S<>0 THEN LET SIN=SIN+1
730 NEXT Z
750 FOR Z=(L AND S=0)+(C AND S<>0) TO 21
755 IF S=0 AND ATTR (Z,C)=56 THEN GO TO 760
756 IF S=0 THEN LET GIU=GIU+1
760 IF S<>0 AND ATTR (L,Z)=56 THEN GO TO 780
762 IF S<>0 THEN LET DES=DES+1
764 NEXT Z
780 LET LATO=SU+GIU+DES+SIN
782 FOR Z=1 TO 4
785 FOR W=1 TO LATO
790 LET R$=R$+("B" AND Z=1)+("S" AND Z=2)+("A" AND Z=3)+("D" AND Z=4)
792 NEXT W: NEXT Z
795 LET G$=""
800 IF SU+GIU<>0 AND C>10 THEN LET R$=R$(SU+1 TO )+R$( TO SU)+" "
801 IF SU+GIU<>0 AND C<10 THEN LET R$=R$(LATO*2+GIU+1 TO )+R$( TO LATO*2+GIU)+" "
802 IF DES+SIN<>0 AND L<10 THEN LET R$=R$(LATO*3+SIN+1 TO )+R$( TO LATO*3+SIN)+" "
803 IF DES+SIN<>0 AND L>10 THEN LET R$=R$(LATO+DES+1 TO )+R$( TO LATO+DES)+" "
815 LET P=0
816 LET C=C+(C>10 AND S=0)-(C<10 AND S=0)
819 LET L=L+(L>10 AND S<>0)-(L<10 AND S<>0)
820 FOR M=1 TO LEN R$-1
825 LET L=L+(R$(1)="B")-(R$(1)="A")
826 LET C=C+(R$(1)="D")-(R$(1)="S")
830 GO SUB 9250
831 LET C1=C: LET L1=L
832 IF R$( TO 2)="BB" AND ATTR (L,C-1)=56 THEN LET C=C-1: GO TO 2500
833 IF R$( TO 2)="SS" AND ATTR (L-1,C)=56 THEN LET L=L-1: GO TO 2500
834 IF R$( TO 2)="AA" AND ATTR (L,C+1)=56 THEN LET C=C+1: GO TO 2500
835 IF R$( TO 2)="DD" AND ATTR (L+1,C)=56 THEN LET L=L+1: GO TO 2500
839 LET G$=G$+R$(1): LET R$=R$(2 TO ): LET P=P+1
840 NEXT M
850 IF P<LATO*2 THEN LET G$=G$+R$(1): GO TO 900
865 LET G$=""
870 FOR G=LEN R$ TO 1 STEP -1
```

Seguito listato 3.

```
875 LET G$=G$+("B" AND R$(G)="A")+("A" AND R$(G)="B")+("D" AND R$(G)="S")+("S" AND R$(G)="D")
880 NEXT G
900 LET E$=E$+G$
905 LET SWITCH=1: LET R$=""
950 GO TO 240
1000 LET E$=E$+("S" AND C<C1 AND C>=10)+("A" AND L<L1 AND L>=10)+("D" AND C>C1 AND C<=10)+("B" AND L>L1 AND L<=10)
1050 RETURN
2000 LET I$=""
2010 FOR K=LEN E$ TO 1 STEP -1
2020 LET I$=I$+("D" AND E$(K)="S")+("S" AND E$(K)="D")+("A" AND E$(K)="B")+("B" AND E$(K)="A")
2030 NEXT K
2040 LET E$=I$
2050 RETURN
2500 LET R$=R$(2 TO ): LET P=P+1: GO TO 850
3000 DATA "A",61,102,103,255,255,103,102,61,"B",0,85,127,193,193,127,85,0
3010 FOR K=1 TO 2: READ W$
3020 FOR Q=0 TO 7: READ F: POKE USR W$+Q,F: NEXT Q
3025 NEXT K
3030 RETURN
9003 LET H=16: LET L2=2: INK 1: PAPER 7
9005 FOR K=1 TO 4
9010 FOR W=0 TO H
9015 PRINT AT L2,L2+W;CHR$ 143
9018 PRINT AT L2+H,L2+W;CHR$ 143
9020 PRINT AT L2+W,L2;CHR$ 143
9023 PRINT AT L2+W,L2+H;CHR$ 143
9025 NEXT W
9040 LET A=H/2+L2
9042 LET B=INT (RND*4): GO TO 9050+B
9050 PRINT AT A,L2; INK 2;"+": GO TO 9080
9051 PRINT AT L2,A; INK 2;"+": GO TO 9080
9052 PRINT AT L2+H,A; INK 2;"+": GO TO 9080
9053 PRINT AT A,L2+H; INK 2;"+"
9080 LET L2=L2+2: LET H=H-4
9081 NEXT K
9082 INK 0: RETURN
9250 FOR W=6 TO 2 STEP -1
9260 LET P(W)=P(W-1)
9270 LET Q(W)=Q(W-1)
9280 NEXT W
9290 LET P(1)=L
9300 LET Q(1)=C
9310 FOR W=6 TO 1 STEP -1
9320 PRINT AT P(W),Q(W);A$(W)
9330 NEXT W
9335 BEEP .05,8: BEEP .05,-5
9340 RETURN
```

Listato 4. *Modifiche necessarie per rendere il listato 3 compatibile allo ZX81. Valgono le considerazioni già fatte per il listato 2. Per lo ZX80 8 Kbyte è necessario aggiungere la linea: 9935 PAUSE 50.*

```
1 REM TOBIA NEL LABIRINTO
     ZX81
2 REM LE SEGUENTI LINEE DEL LISTATO SPECTRUM NON VANNO USATE:
70,720,3000,3010,3020,3025,3030
9335
80 LET T=0
81 LET R$=""
82 LET E$=""
83 LET SWITCH=0
150 LET A$="▒OOOO "
234 IF L=Y AND C=X THEN GOSUB 1000
```

```
235 IF L=Y AND C=X THEN GOTO 600
236 IF PEEK (PEEK 16398+256*PEEK 16399)=128 THEN GOTO 700
245 IF SWITCH=0 THEN LET R$=E$
246 IF SWITCH=0 THEN LET SWITCH=1
336 LET C1=C
338 LET L1=L
600 PRINT AT Y-1,X;"▀▀"
610 LET T=T+1
611 IF T>2 THEN LET T=1
612 IF T=2 THEN GOTO 620
```

Seguito listato 4.

```
613 LET X=26
614 LET Y=10
615 PRINT AT Y,X;"*"
616 GOSUB 2000
617 LET SWITCH=0
618 LET R$=""
619 GOTO 220
620 LET Y=10
621 LET X=10
622 PRINT AT Y,X;"*"
623 GOSUB 2000
624 LET SWITCH=0
625 LET R$=""
626 GOTO 220
700 PRINT AT L,C-1;
701 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
702 LET S=0+(1 AND ATTR=128)
703 PRINT AT L,C+1;
704 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
705 LET S=S+(1 AND ATTR=128)
706 LET SU=0
707 LET GIU=0
708 LET DES=0
709 LET SIN=0
716 PRINT AT Z,C;
718 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
719 IF S=0 AND ATTR=0 THEN GOTO
750
722 PRINT AT L,Z;
723 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
724 IF S<>0 AND ATTR=0 THEN GOT
O 750
725 IF S<>0 THEN LET SIN=SIN+1
751 PRINT AT Z,C;
752 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
753 IF S=0 AND ATTR=0 THEN GOTO
780
755 IF S=0 THEN LET GIU=GIU+1
756 PRINT AT L,Z;
760 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
762 IF S<>0 AND ATTR=0 THEN GOT
O 780
764 IF S<>0 THEN LET DES=DES+1
765 NEXT Z
792 NEXT W
793 NEXT Z
831 LET C1=C
832 LET L1=L
833 GOTO 2800
834 LET G$=G$+R$(1)
835 LET R$=R$(2 TO )
```

Seguito listato 4.

```
839 LET P=P+1
850 IF P<LATO*2 THEN LET G$=G$+
R$(1)
851 IF P<LATO*2 THEN GOTO 900
905 LET SWITCH=1
906 LET R$=""
2500 LET R$=R$(2 TO )
2502 LET P=P+1
2504 GOTO 850
2800 PRINT AT L,C-1;
2802 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
2805 IF R$( TO 2)="BB" AND ATTR=
21 THEN GOTO 2900
2810 PRINT AT L-1,C;
2812 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
2815 IF R$( TO 2)="SS" AND ATTR=
21 THEN GOTO 2910
2820 PRINT AT L,C+1;
2822 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
2825 IF R$( TO 2)="AA" AND ATTR=
21 THEN GOTO 2920
2830 PRINT AT L+1,C;
2832 LET ATTR=PEEK (PEEK 16398+2
56*PEEK 16399)
2835 IF R$( TO 2)="DD" AND ATTR=
21 THEN GOTO 2930
2899 GOTO 834
2900 LET C=C-1
2901 GOTO 2500
2910 LET L=L-1
2911 GOTO 2500
2920 LET C=C+1
2921 GOTO 2500
2930 LET L=L+1
2931 GOTO 2500
9000 LET H=16
9001 LET L2=2
9015 PRINT AT L2,L2+W;"■"
9018 PRINT AT L2+H,L2+W;"■"
9020 PRINT AT L2+W,L2;"■"
9023 PRINT AT L2+W,L2+H;"■"
9042 LET B=INT (RND*4)
9043 GOTO 9050+B*2
9050 PRINT AT A,L2;"+"
9051 GOTO 9080
9052 PRINT AT L2,A;"+"
9053 GOTO 9080
9054 PRINT AT L2+H,A;"+"
9055 GOTO 9080
9056 PRINT AT A,L2+H;"+"
9080 LET L2=L2+2
9081 LET H=H-4
9082 NEXT K
9084 RETURN
```

Listato 5. *Tiro mancino: è un gioco che si svolge su una scacchiera a nove posizioni. Ogni giocatore dispone di tre pedine e può muovere solo verso l'alto e mangiare solo in senso obliquo. Perde chi ad un certo punto non può più muovere. Ogni partita si esaurisce in poche mosse, e giocando, il computer diventa più abile.*
Il listato presentato si riferisce allo ZX81 ma il programma gira su Spectrum e ZX80 8 Kbyte senza modifiche.

```
   4 REM TIRO MANCINO
  10 GOSUB 9000
  12 PRINT "QUESTO GIOCO, CHE SE
MBRA SEMPLI-CE, E INVECE ABBASTA
NZA COMPLES-SO"
  13 PRINT
  14 PRINT "LE  G  SONO LE TUE P
EDINE"
  15 PRINT "LE  S  SONO QUELLE D
EL COMPUTER"
  16 PRINT "I  *  SONO SPAZI VU
OTI"
```

```
  18 PRINT
  20 PRINT "PUOI MUOVERE SOLO VE
RSO L ALTO  VERTICALMENTE E MANG
IARE IN SEN-SO OBLIQUO VERSO L A
LTO"
  23 PRINT
  25 PRINT "PERDE CHI SI TROVA N
ELLA CONDI- ZIONE DI NON POTER M
UOVERE PIU"
  26 PRINT AT 21,0;"PREMI UN TAS
TO"
  28 PAUSE 20000
```

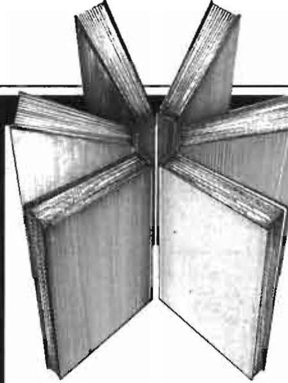
Seguito listato 5.

```
  30 GOSUB 9000
  32 PRINT "PER FARE LA MOSSA DE
VI INDICARE DUE NUMERI:"
  34 PRINT "LA CASELLA DI PARTEN
ZA"
  35 PRINT "E QUELLA DI ARRIVO"
  37 PRINT
  39 PRINT "LA SCACCHIERA E DISE
GNATA A DE- STRA. SULLA SINISTRA
 PUOI VEDEREI NUMERI DI CASELLA
PER TUA COMODITA"
  41 PRINT AT 21,0;"PREMI UN TAS
TO"
  45 PAUSE 20000
  50 GOSUB 9000
  90 DIM A(100)
 100 DIM B(9)
 120 DIM C(5)
 125 LET E=-1
 130 LET D=1
 210 FOR K=1 TO 3
 220 LET B(K)=E
 225 LET B(K+3)=0
 240 LET B(K+6)=D
 250 NEXT K
 260 GOSUB 1000
 275 LET K1=0
 305 PRINT "BATTI LA MOSSA (DA..
A..)"
 310 INPUT MOS
 320 LET N=INT (MOS/10)
 325 LET CON=D
 332 LET Z=MOS-N*10
 350 GOSUB 8200
 360 IF SB<>0 THEN GOTO 310
 368 LET B(N)=0
 380 LET B(Z)=D
 390 GOSUB 1000
 410 LET CON=E
 420 GOSUB 6000
 430 IF WD=0 THEN GOTO 600
 440 PRINT "PREMI UN TASTO PER M
E"
 450 PAUSE 20000
 460 GOSUB 7000
 470 LET B(N)=0
 480 LET B(Z)=E
 490 GOSUB 1000
 510 LET CON=D
 520 GOSUB 6000
 530 IF WD>0 THEN GOTO 300
 536 PRINT
 540 PRINT "H O    V I N T O   I
O"
 550 GOTO 620
 600 GOSUB 8000
 614 PRINT
 615 PRINT "          H A I    V
I N T O"
 620 PRINT
 630 PRINT "UN TASTO PER CONTINU
ARE"
 640 PAUSE 20000
 650 GOTO 200
 660 STOP
1000 GOSUB 9000
1020 FOR K=0 TO 2
1030 LET X=3*K+1
1040 PRINT TAB 8;X;" ";X+1;" ";X
+2;TAB 16;
1050 FOR Y=0 TO 2
1060 LET O=B(X+Y)
1070 IF O=0 THEN PRINT " *";
1080 IF O=D THEN PRINT " G";
1090 IF O=E THEN PRINT " S";
1100 NEXT Y
1110 PRINT
1120 PRINT
1130 NEXT K
1140 PRINT
```

Seguito listato 5.

```
1150 RETURN
6000 LET WD=0
6020 FOR N=1 TO 9
6030 IF NOT (B(N)=CON) THEN GOTO
 6100
6040 FOR Z=1 TO 9
6050 GOSUB 8200
6060 IF SB>0 THEN GOTO 6090
6070 LET C(WD+1)=Z+10*N
6080 LET WD=WD+1
6090 NEXT Z
6100 NEXT N
6110 RETURN
7000 LET WK=0
7020 FOR K=1 TO 9
7030 LET WK=B(K)+1+3*WK
7040 NEXT K
7060 FOR K=1 TO 100
7070 LET K1=K
7080 IF A(K1)=0 THEN GOTO 7400
7090 IF A(K1)=WK THEN GOTO 7200
7095 NEXT K
7200 FOR K=1 TO 10
7210 IF A(K1+K)>-1 THEN GOTO 724
0

7230 NEXT K
7240 IF K=1 THEN GOTO 7300
7250 LET WK=WK+INT ((K-1)*RND)+1
7260 LET N=INT (-A(K1)/10)
7270 LET Z=-A(K1)-10*N
7280 LET JH=K1
7290 RETURN
7300 GOSUB 8000
7310 LET JH=100
7330 LET WD=INT (WD*RND)
7340 LET N=INT (C(WD+1)/10)
7350 LET Z=C(WD+1)-10*N
7360 RETURN
7400 LET A(K1)=WK
7420 FOR K=1 TO WD
7430 LET A(K1+K)=-C(K)
7440 NEXT K
7450 LET K1=K1+INT (1+WD*RND)
7460 GOTO 7250
8000 IF JH=100 THEN GOTO 8050
8010 FOR V=JH TO 99
8030 LET A(V)=A(V+1)
8040 NEXT V
8050 LET A(100)=0
8060 RETURN
8200 LET SB=1
8220 IF N>9 OR Z>9 OR N<1 OR Z<1
 THEN RETURN
8230 IF B(Z)=CON THEN RETURN
8240 IF NOT (B(N)=CON) THEN RETU
RN
8250 IF (INT ((N-1)/3)-INT ((Z-1
)/3))<>CON THEN RETURN
8260 LET R=ABS (N-Z)
8270 IF R=3 AND B(Z)=0 THEN GOTO
 8300
8280 IF (R=2 OR R=4) AND B(Z)=-C
ON THEN GOTO 8300
8290 RETURN
8300 LET SB=0
8310 RETURN
9000 CLS
9010 PRINT "00000000000000000000
000000000000"
9011 PRINT
9020 PRINT "            TIRO  MANCI
NO"
9021 PRINT
9030 PRINT "00000000000000000000
000000000000"
9040 PRINT
9050 PRINT
9059 RETURN
```



2+2=APPLE



Due Riviste famose, specializzate, informatissime

BIT - PERSONAL SOFTWARE

Due volumi preziosi per chi vuole approfondire la conoscenza del suo computer

INTERFACCIAMENTO DELL'APPLE
196 pagine
Cod. **334B**
Lire 14.000

APPLE II Guida all'uso
390 pagine
Cod. **331P**
Lire 26.000

Una sola firma prestigiosa per chi si interessa di informatica e di elettronica

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

COUPON D'INFORMAZIONE

Desidero ricevere un numero omaggio di ☐ BIT - ☐ PERSONAL SOFTWARE insieme a maggiori informazioni sulle condizioni di abbonamento

INVIATEMI CONTRASSEGNO

n° copie	codice	Prezzo unitario	Prezzo totale
	334B	**L. 14.000**	
	331P	**L. 26.000**	
contributo fisso spese di spedizione			L. 2000
Totale			

Nome

Cognome

Via

Cap Città Prov.

Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A.

nuovidea

PERSONAL SOFTWARE

Superman

Un gioco sullo Spectrum anche per chi non vuole usare il linguaggio macchina

di *Ivano Parbuono*

Superman è un gioco scritto completamente in BASIC, non può quindi avere la grafica o la velocità possibile con il linguaggio macchina, ma non è necessario essere assembleromani per scrivere un buon gioco.

La figura 1 visualizza la situazione iniziale del gioco, mentre in figura 2 potete trovare lo schema a blocchi.

Ecco la descrizione dei punti più importanti del programma riportato nel listato 1. La linea 50 serve ad azzerare il totalizzatore del punteggio massimo; dalla linea 100 alla 178 si trova una routine che definisce caratteri grafici che devono essere inseriti in ordine alfabetico dalla A alla I compresa.

La linea 180 azzera il punteggio della partita ed inserisce il contatempo che inizia da 60 e viene decrementato fino a 0 dalla linea 650 una volta lanciato il programma.

Dalla linea 300 alla linea 396 viene creato il quadrato del gioco e i grafici interni ad esso, mentre le illustrazioni che vanno dalla linea 400 alla 470 fanno sì che Superman possa essere spostato per mezzo dei tasti 5-6-7-8 in qualsiasi direzione dando così la possibilità di uccidere l'alieno che si muove per mezzo della funzione RND alla linea 690.

Quando Superman uccide l'alieno sul video appare una croce e il totalizzatore viene incrementato di 10 punti alla linea 2000.

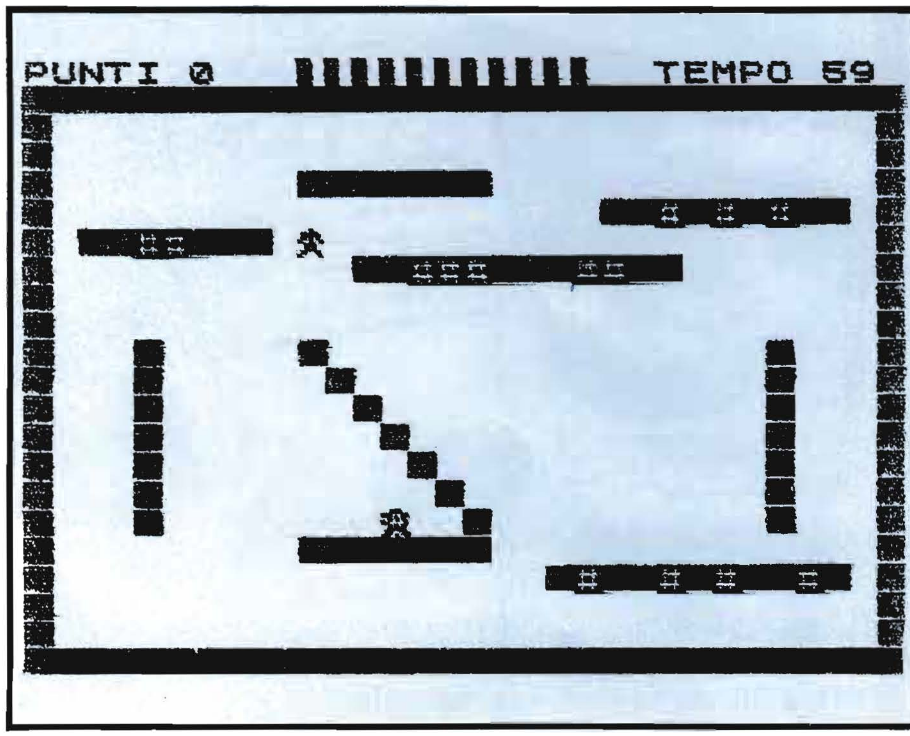


Figura 1. *Situazione iniziale del gioco: Superman (in alto) deve catturare l'alieno (in basso). Questi può muovere anche attraverso il muro in diagonale.*

La linea 2500 serve nel caso sia scaduto il tempo e pulisce lo schermo per fare spazio al punteggio appena totalizzato e al massimo punteggio realizzato nelle partite precedenti. Se si è realizzato un punteggio superiore a 100 la linea 2850 rimanda alla subroutine dalla linea 3000 alla 3300 che suona una musica trionfale e fa apparire Superman ingrandito al centro dello schermo terminando il gioco.

Questo programma è stato predisposto per essere giocato anche con joystick Kempston e la linea 475 ha il compito di fare in modo che vengano seguiti i controlli del joystick. ■

```
   5 REM © IVANO  PARBUONO
   6 BEEP 1,5: BEEP 2,12
   7 PRINT AT 2,6; INK 1;"▄▄▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
   8 PRINT AT 3,8; INK 2;"S U P
E R M A N": BEEP .6,5: BEEP 2,24
   9 PRINT AT 4,8; INK 3;"▀▀▀▀▀▀
▀▀▀▀▀▀▀▀▀▀▀"
  10 BEEP .8,24: BEEP 2,6: INK 1
: PRINT AT 8,3;"████████████████
█████████": INK 2: PRINT AT 9,3;
" IDEATO E REALIZZATO DA
    PARBUONO   IVANO
   Via A. di CAMBIO 4
       VERONA": INK 1: PRINT "
  ██████████████████████████"
  12 BEEP 1,12: BEEP .9,24: BEEP
 1,9: PRINT AT 18,2; INK 4;"SI
PUO' GIOCARE  ANCHE  CON
JOYSTICK  KEMPSTON": BEEP 1,12:
BEEP 2,2: BEEP .9,24: PRINT AT 2
0,1; INK 5;"▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪
▪▪▪▪▪▪▪▪▪▪▪▪": BEEP 1,12: BEEP 1,
24: PAUSE 150
  50 LET vi=0
 100 DATA BIN 00011100,BIN 11100
011,BIN 00101010,BIN 11000001,BI
N 00101010,BIN 00111110,BIN 0110
0011,BIN 11001001
 110 DATA BIN 00010000,BIN 01111
100,BIN 01010100,BIN 10010010,BI
N 00111000,BIN 00101000,BIN 0100
0100,BIN 10000010
 120 DATA BIN 00010000,BIN 00010
000,BIN 00010000,BIN 01111100,BI
N 00010000,BIN 00010000,BIN 0001
0000,BIN 00010000
 125 DATA BIN 00000011,BIN 00000
011,BIN 00111111,BIN 00110011,BI
N 00110011,BIN 01000011,BIN 1100
0011,BIN 10000011
 130 DATA BIN 11000000,BIN 11000
000,BIN 11111100,BIN 11001100,BI
N 11000010,BIN 11000011,BIN 1100
0001,BIN 11000001
 135 DATA BIN 00000011,BIN 00011
111,BIN 00011010,BIN 00011000,BI
N 00110000,BIN 01100000,BIN 0110
0000,BIN 11100000
```

Listato 1. Listato del programma Superman privo della sezione in linguaggio macchina.

Seguito listato Superman

```
 140 DATA BIN 11000000,BIN 11111
000,BIN 01011000,BIN 00011000,BI
N 00001100,BIN 00000110,BIN 0000
0110,BIN 00000111
 150 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 160 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 170 FOR f=0 TO 7: READ a: POKE
USR "†"+f,a: NEXT f
 172 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 174 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 176 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 178 FOR f=0 TO 7: READ a: POKE
USR "■"+f,a: NEXT f
 180 LET pu=0: LET tp=60
 200 BORDER 2: PAPER 6: INK 0: C
LS
 210 LET d=6: LET e=10: LET f=16
: LET g=14
 220 LET a$="■"
 230 LET o$=""
 300 PRINT AT 1,0; INVERSE 1; IN
K 0;"________________________________"
 310 FOR a=1 TO 20
 320 PRINT INVERSE 1; INK 0;"_";
AT a,31;"_"
 330 NEXT a
 340 PRINT INVERSE 1; INK 0;"________________________________"
 350 PRINT AT 0,0;"PUNTI ";pu; I
NK 2;"   ▮▮▮▮▮▮▮▮▮▮▮▮"
 370 PRINT AT 6,2; INVERSE 1; IN
K 5;"__##____"
 375 PRINT AT 7,12; INVERSE 1; I
NK 2;"__###____##__"; INK 5;AT 5,
21;"___#_#_#__"
 380 PRINT AT 16,19; INVERSE 1;
INK 2;"_#___#_#___#_"
 394 FOR b=10 TO 16
 395 PRINT AT b,4; INVERSE 1; IN
K 4;"_";AT b,27;"_"; INK 2;AT 17
,b;"_"; INK 4; INK 5;AT b,b;"_";
 INK 2;AT 4,b;"_"
 396 NEXT b
 400 PRINT AT d,e;" "
 450 LET o$=INKEY$
 460 LET d1=d: LET e1=e
 470 LET d=d+(o$="6")-(o$="7"):
LET e=e+(o$="8")-(o$="5")
 475 LET d=d+(IN 31=4)-(IN 31=8)
: LET e=e+(IN 31=1)-(IN 31=2)
 510 IF SCREEN$ (d,e)="" THEN GO
 TO 2000
 520 IF SCREEN$ (d,e)="_" THEN L
ET d=d1: LET e=e1: LET o$="": GO
 TO 460
 560 PRINT AT d,e; INK 2;a$
 610 IF a$="" THEN LET c$=d$
 650 PRINT AT 0,23;"TEMPO ";INT
tp;" ": LET tp=tp-.2: IF tp<0 TH
EN GO TO 2500
 660 BEEP .008,5
 670 PRINT AT f,g;" "
 680 LET f1=f: LET g1=g
 690 LET f=f+INT (RND*3)-1: LET
g=g+INT (RND*3)-1
 700 IF SCREEN$ (f,g)="_" THEN L
ET f=f1: LET g=g1: GO TO 680
 720 IF SCREEN$ (f,g)="" THEN GO
 TO 2000
 750 PRINT AT f,g; INK 1;"■"
 780 GO TO 400
2000 PRINT AT f,g;"†": BEEP .5,1
: LET pu=pu+10
2020 PRINT AT 0,0;"PUNTI ";pu
2030 BEEP .8,5
2100 GO TO 200
2500 PRINT AT 0,15; PAPER 3;"TEM
PO 0": BEEP 1,5: BEEP .9,10: CLS

2600 PRINT AT 6,3 ;"HAI TOTALIZZ
ATO ";pu; " PUNTI"
2700 IF pu>vi THEN LET vi=pu
2800 PRINT AT 12,0; PAPER 6;"IL
PUNTEGGIO MAX E' DI ";vi;" PUNTI
": PAUSE 100
2850 IF pu>100 THEN GO SUB 3000
2900 INPUT "        PREMI ENTER
PER              GIOCARE ANCOR
A "; LINE o$: GO TO 180
3000 FLASH 1: BORDER 1: PAPER 4:
 INK 2: BEEP 1,25: BEEP 3,1: BEE
P .9,19: BEEP 1,9: BEEP .04,35:
BEEP 1,2
3100 FLASH 0: CLS : BORDER 2: PA
PER 6: FLASH 1: INK 1: PRINT AT
9,5;"BRAVO  BRAVO  SEI  UN": FLA
SH 1: PRINT AT 10,8;"SUPER  SUPE
RMAN": FLASH 0
3200 FLASH 1: INK 2: PRINT AT 12
,14;"■■"
3250 FLASH 1: INK 2: PRINT AT 13
,14;"■■"
3300 BEEP .07,1: BEEP 2,4: BEEP
1,9: BEEP 1,9: BEEP .09,12: BEEP
 2,5
3500 FLASH 0: GO TO 2900
```

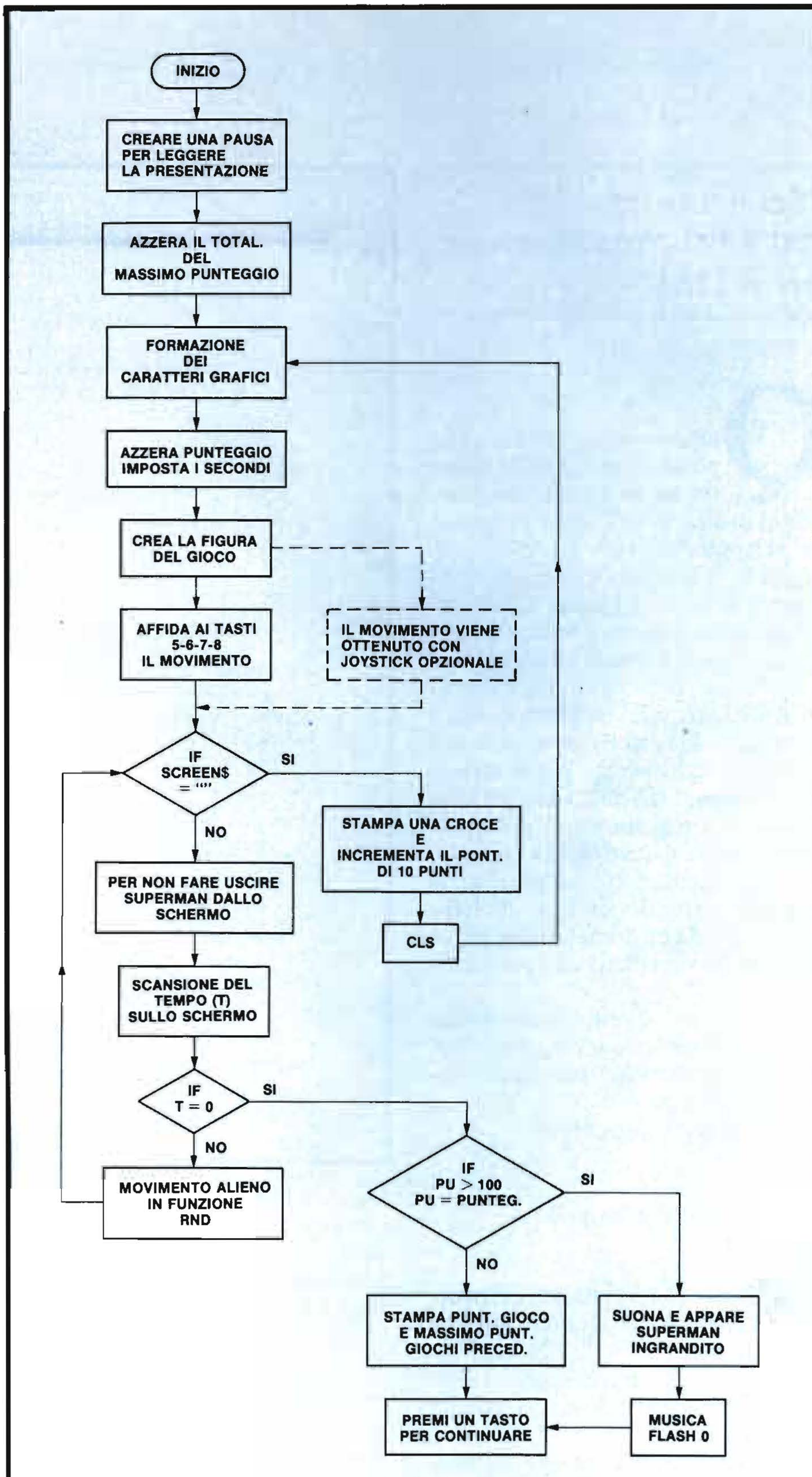


Figura 2. *Diagramma a blocchi del programma Superman.*

Un potente word processor

"Screen oriented" text editor professionale per il DAI

di *Giulio Morpurgo*

Questo programma, che possiede molte delle caratteristiche tipiche dei text-editor di alta classe, è stato progettato per poter scrivere un testo, correggerlo, e stamparlo in modo ordinato e con poca fatica. L'uso del programma è facile e veloce da imparare, e non richiede alcuna abilità particolare. Il programma è scritto in BASIC (tranne un certo numero di routine Assembler necessarie per rendere soddisfacente la velocità di esecuzione), e ciò ha un duplice vantaggio: il programma è facilmente modificabile e adattabile a casi particolari necessari all'utente, e, nella maggior parte dei casi, un disastro provocato da un comando sbagliato può essere rimediato con poca fatica.

Spero che questo programma possa tornare utile a chi, come me, possiede questo bel personal computer che ha però il difetto di avere poco software disponibile.

```
           ======================
           |  INIZIALIZZAZIONE  |
           ======================
                     |
|------------------->|<---------------------------|
|                    |                            |
|                    V                            |
|          =====================                  |
|          | MODULO DI COMANDO |L,S  ----------   |
|          |                   |------|esegui |-->|
|          |                   |M,P,T ---------   |
|          |                   |                  |
|          =====================                  |
|          A,I|       |    F,X|          / \      |
|             |<--|   |       |         /   \ no  |
|        |----|   |   |       |------->/trovi\--->|
|   -----V----    |   |                 \ ? /
|   | MODO A |    |   |<------------|    \ /
|   |--------| |--|   |             |    \ /
|   | MODO I | |      V             |     |si
|   ---------- |  ===========       |<-----|
|    #A|   $|--|  |  MODO   |       |
|     #|          | CURSORE |       |
|      |          ===========       |
|      V         #|  ^| $| -----    |
|-----------------|   |  |-|ese|-->|
                      |    |gui|    |
                      V    -----    |
                ==============      |
                | OPERAZIONI |      |
                | CON  I     |----->|
                | MARCHI     | E    |
                ==============      |
                >,<,Z | I,P,S,DEL   |
                      |             |
                      V             |
                  -----------       |
                  | esegui   |------|
                  -----------
```

Figura 1. *La struttura del programma. Si passa da un blocco all'altro battendo i comandi indicati a fianco dei rami del grafo.*
Note:
$ = *qualsiasi altro carattere.*
Esegui = esegui il comando corrispondente al carattere battuto.

Caratteristiche principali del programma

— Dimensione massima del testo: oltre 22000 caratteri (circa 11 pagine dattiloscritte);

— Possibilità di correggere il testo posizionandosi sugli errori con un cursore;

— Ricerca e sostituzione automatica di stringhe di caratteri nel testo;

— Possibilità di introdurre nel testo caratteri ASCII non presenti sulla tastiera;

— Possibilità di definire una parte del testo tramite due puntatori ("marchi") iniziale e finale, e di spostare, cancellare, stampare, sostituire questa parte di testo;

— Controllo automatico di "buffer pieno", con preavviso quando mancano 60 caratteri al riempimento;

— Uso reso più facile della diversa colorazione che assume lo schermo a seconda delle operazioni che si

stanno compiendo e dalla linea di comando in fondo allo schermo;

— Stampa dei testi marginata a destra e sinistra, ottenuta sia variando le dimensioni degli spazi tra le parole, sia suddividendo le parole in sillabe per andare a capo.

Struttura del programma

In figura 1 è rappresentata graficamente la struttura text-editor; terminata l'INIZIALIZZAZIONE, il programma entra nel MODO COMANDO, dal quale può eseguire un certo numero di comandi (SAVE, LOAD, PRINT...) e rimettersi a disposizione del prossimo comando, oppure può entrare in uno dei sottomodi MODO A, INSERIMENTO TESTO NUOVO, MODO CURSORE, ciascuno con le sue caratteristiche, dai quali si ritorna al MODO COMANDO battendo il carattere "#". Il MODO CURSORE si raggiunge anche tramite l'esecuzione dei comandi FIND e FIND & SUBSTITUTE; in esso sono disponibili tutti i comandi necessari per muovere il cursore sul video e correggere il testo, e da esso si può passare al modo OPERAZIONI CON I MARCHI, che consente di definire e spostare una parte del testo.

I "modi di operazione"

I diversi modi di operazione sopra elencati si differenziano tra di loro per le funzioni che compaiono; inoltre il riconoscimento del modo in cui si trova il programma in un certo momento è reso più facile sia dalla diversa colorazione che assume lo schermo, sia da alcune scritte che compaiono nell'ultima riga dello schermo. (Questa ultima riga ha la funzione di mantenere il dialogo con l'utente durante quelle funzioni che richiedono la visualizzazione del testo sul video). In tabella 1 sono elencate le associazioni tra modi di operazione e colori dello schermo.

MODO	COL. SFONDO	COL. TESTO
COMANDO	Bianco	Nero
CREAZIONE NUOVO TESTO	Verde	Nero
MODO A	Giallo	Nero
CURSORE	Azzurro	Nero
OPERAZIONI CON I MARCHI	Nero	Bianco

Tabella 1. *Associazione tra i colori e i modi di operazione.*

```
Si puo' sempre dividere una parola

a) tra due consonanti uguali consecutive
b) dopo la "c" seguita da "q"
c) prima di una "s" non doppia
d) dopo una "l","m","n" o "r" seguita
   da una altra consonante
e) prima di una consonante semplice
   (cioe' preceduta e seguita da vocali)
```

Tabella 2. *Regole per la divisione delle parole in sillabe.*

Descrizione dei comandi

Esaminiamo ora i diversi comandi e le loro funzioni.

Per comodità, conviene dividere i comandi in tre categorie, corrispondenti al modo in cui ci si trova quando il comando viene impartito.

Comandi nel modo comando

L (LOAD): Permette di caricare in memoria un testo dalla cassetta.

S (SAVE): Permette di memorizzare il testo sulla cassetta, eventualmente assegnandogli un nome. Le dimensioni del file memorizzato dipendono dalla lunghezza del testo, perché il programma, prima di salvare la matrice TESTO%, ne modifica la dimensione (linea 1209).

M (SHOW): Mostra il numero di byte ancora liberi per il testo.

P (PRINT): Permette di stampare il testo in memoria. Sono disponibili

due tipi di stampa: quella normale e la STAMPA MARGINATA.

La stampa normale serve essenzialmente per rivedere il testo sul video, senza impaginarlo in modo ordinato. Lo scorrimento del testo può essere interrotto battendo uno "space", e riprenderà premendo un qualsiasi carattere.

La stampa marginata è invece molto più sofisticata: essa stampa il testo in modo che risulti sempre allineato a destra, variando la larghezza degli spazi tra le parole (se una riga inizia con due o più spazi iniziali, questi non vengono alterati). Poiché la stampante a mia disposizione è grafica, la minima variazione possibile è di 1/7 di spazio. In questo modo è possibile mantenere gli interspazi tra le parole uguali nell'ambito della stessa riga. Inoltre, nel caso che si dovesse variare di troppo la spaziatura per allineare il testo senza spezzare una parola, il programma è dotato della capacità di andare a capo correttamente, spezzando le parole in sillabe, seguendo le regole elencate in tabella 2.

Prima di stampare il testo si possono definire:
— il numero di colonne,
— la spaziatura tra le righe,
— se si scrive su fogli singoli o su modulo continuo,
— il numero di righe in una pagina (dopo il quale occorre dare un carattere di next-page),
— il margine da lasciare a sinistra del testo.

I (INSERT): Questo comando provoca l'ingresso nel modo INSERIMENTO TESTO NUOVO. Il video viene cancellato, e i puntatori reinizializzati (il vecchio testo viene perso, ed è recuperabile solo interrompendo l'esecuzione con un BREAK, e ripristinando il vecchio valore di MAXPUNT%). Tramite la tastiera è ora possibile inserire il nuovo testo. Per uscire da questo modo, e tornare al MODO COMANDO, occorre battere un carattere "#".

Per inserire nel testo un carattere

```
TEXBUF% : indirizzo iniziale del buffer
          del testo (e' costante).
MAXPUNT% : ultimo byte occupato nel buffer
           del testo.
TEXTMAX% : indirizzo finale del buffer
           del testo
PUNT% : puntatore al buffer di testo : e'
        usato nel MODO CURSORE , e punta
        al byte dove si trova il carattere
        nella posizione del cursore.
RISBUF% : indirizzo iniziale del buffer
          dove vengono inseriti i caratteri
          nel MODO A, prima di essere tra-
          sferiti nel buffer del testo
RISPUNT% : puntatore al buffer suddetto
MAXBUF%  : indirizzo finale del buffer
SEARCHBUF% :indirizzo iniziale del buffer
            in cui e' memorizzata la strin-
            ga da cercare.
SOSBUF% : indirizzo iniziale del buffer in
          cui e' memorizzata la stringa da
          sostituire.
WBUF% : indirizzo iniziale del buffer dove
        sono costruite le righe del testo
        durante la stampa marginata.
WPUNT% : puntatore al buffer del testo du-
         rante la stampa marginata

COL%,ROW% : posizioni x e y del cursore
            durante il MODO CURSORE.

INIMARK%  : valore marco iniziale
FINMARK%  : valore marco finale
LASTMARK% : flag : =1 se l'ultimo marco
            settato e' INIMARK%
                   =2 se e' FINMARK%
                   =0 se i due marchi non
            sono settati.
```

Tabella 3. *Alcune variabili usate dal programma.*

ASCII non presente sulla tastiera del DAI, battere SHIFT FRECCIA A SINISTRA; ora introdurre il codice del carattere in questione battendo i tasti numerici, e poi battere di nuovo SHIFT FRECCIA A SINISTRA: il carattere sarà inserito nel testo. L'ultimo carattere introdotto così viene memorizzato, e, per introdurlo nuovamente, basterà battere due volte di seguito il tasto SHIFT FRECCIA A SINISTRA.

Questa operazione è disponibile anche nel MODO A e nel MODO CURSORE.

A (ADD): Provoca l'ingresso nel MODO A, che consente di inserire, attraverso la tastiera, un nuovo testo dentro a quello già esistente, partendo dalla posizione attuale del cursore. Al momento dell'esecuzione del comando, tutto il testo visualizzato oltre la posizione del cursore viene cancellato dal video, per la-

```
byte 7  : 1=consonante
byte 6  : 1= L, M, N o R
byte 5  : carattere mai ultimo di riga
byte 4  : carattere mai primo di riga
byte 3  : non usato
byte 2  : 1=carattere stretto
byte 1  : 1=carattere largo
byte 0  : 1=carattere di interpunzione
```

Tabella 4. *Codifica delle caratteristiche dei caratteri nella tabella CHAR.*

Listato 1. *Il programma completo di editor e stampa nella versione per stampanti grafiche.*

```
10    REM ****************************************
14    REM *                                      *
15    REM *  DAI  SCREEN  ORIENTED  TEXT  EDITOR  *
16    REM *                                      *
17    REM *     by    GIULIO  MORPURGO           *
18    REM *                                      *
20    REM ****************************************
100   REM *******     INIZIALIZZAZIONE    *******
110   MODE 0:CLEAR 26000
111   POKE #FF05.#40:POKE #131.1:PRINT CHR$(12);
119   DIM ASSEMB%(128.0.1.0)
120   DIM TESTO%(99.0.57.0)
121   DIM OFSET%(41.0,4.0),DOD%(40.0)
125   GOSUB 30000
129   WBUF%=25700
130   RISBUF%=25000:MAXBUF%=25500:TEXBUF%=#702:TEXTMAX%=24900
131   FLAG%=0
132   SEARCHBUF%=25550:SOSBUF%=25600
133   MAXPUNT%=TEXBUF%:PUNT%=TEXBUF%
134   COL%=0:ROW%=23:MAXCOL%=60
135   GOSUB 10500
140   EXI$="    TO EXIT : TYPE #"
200   GOTO 1995
1000  REM ------------------
1001  REM INSERIMENTO DI UN TESTO NUOVO
1002  COLORT 13 0 0 0
1003  RITX=1
1005  MAXPUNT%=TEXBUF%:PRINT CHR$(12);:GOSUB 10500
1006  COL%=0:ROW%=23:MAXCOL%=60
1007  GOSUB 10900:PRINT "INSERIMENTO TESTO NUOVO";EXI$;
1008  CURSOR 0.23
1010  A=GETC:IF A=0.0 THEN 1010
1015  IF A=22.0 THEN GOSUB 1300
1020  IF A=ASC("#") THEN PUNT%=MAXPUNT%:RT%=1:GOTO 1995
1030  POKE MAXPUNT%,A:MAXPUNT%=MAXPUNT%+1.0:JPUNT%=MAXPUNT%:GOTO 12000
1031  COL%=COL%+1
1035  PUNT%=MAXPUNT%:COLORT 13 0 0 0:GOSUB 10500
1040  IF A=13 THEN PRINT CHR$(2);:COL%=0:ROW%=ROW%-1
1050  PRINT CHR$(A);
1055  IF A=8.0 THEN MAXPUNT%=MAXPUNT%-2:COL%=COL%-2
1056  IF COL%<0 THEN COL%=COL%+1:MAXPUNT%=MAXPUNT%+1:GOTO 1056
1060  IF COL%=MAXCOL% THEN PRINT :COL%=0:ROW%=ROW%-1
1065  IF ROW%=0 THEN ROW%=1:GOSUB 10600:CURSOR COL%,1
1070  GOTO 1010
1200  REM ------------------------------
1205  REM SALVATAGGIO DI UN FILE
1206  POKE TEXBUF%-3,MAXPUNT% MOD 256:POKE TEXBUF%-2,MAXPUNT%/256
1207  B9%=PEEK(#6F9):BA%=PEEK(#6FA)
1208  PRINT CHR$(12);
1210  PRINT "SALVO IL FILE SU NASTRO:DAMMI IL NOME"
1220  INPUT FILENAME$:PRINT
1230  PRINT "PREPARA IL REGISTRATORE E AVVIALO"
1234  PRINT "SE SEI PRONTO BATTI UN TASTO"
1235  B=GETC:IF B=0.0 THEN 1235
1236  D%=MAXPUNT%-TEXBUF%+60:POKE #6F9,D%/256:POKE #6FA,D% MOD 256
1240  SAVEA TESTO% FILENAME$
1245  POKE #6F9,B9%:POKE #6FA,BA%
1250  GOSUB 4000:GOTO 1995
1300  REM ---CARATTERI SPECIALI---
1305  B%=0
1310  GOSUB 10400:IF A=22 THEN 1330
1315  IF A<48 OR A>57 THEN 1310
1320  B%=B%*10+A-48:GOTO 1310
1330  IF B%=0 THEN A=OLDSPECIAL:RETURN
1335  A=B%:OLDSPECIAL=A:RETURN
1400  REM ------------------------
1405  REM STAMPA DEL TESTO
1406  POKE #131,1:COLORT 8 0 0 0:PRINT CHR$(12);:GOSUB 10500
1407  POKE #FF05,8
1410  INPUT "NUMERO COLONNE";N%:PRINT
1412  INPUT "STAMPANTE <1=SI/0=NO>";F%:PRINT
1413  IF F%=0 THEN 1420
1414  INPUT "STAMPA MARGINATA <1=SI/0=NO>";F1%:PRINT
1415  INPUT "NUMERO RIGHE";MAXRIG%:RIGA%=0:PRINT
1416  INPUT "SPAZIATURA";SPAZ%:PRINT
1417  INPUT "FOGLI SINGOLI <0=SI/1=NO>";AUTOFORM%:PRINT
```

sciare spazio al nuovo testo. Il vecchio testo sarà poi inserito in coda al nuovo quando si uscirà, tramite il comando #A, dal MODO A.

T (TOP OF FILE): Il puntatore al testo viene riportato all'inizio, e sul video viene visualizzata la prima "pagina" di testo. Questo comando è molto utile quando, dopo aver inserito un testo, lo si voglia percorrere dall'inizio con il cursore per posizionarsi sui punti da correggere, e per ripartire dopo una interruzione del programma volontaria o no.

F (FIND): Consente di cercare una stringa di caratteri all'interno del testo. La ricerca inizia dal valore attuale del puntatore al testo. Se la stringa che si cerca esiste, il puntatore si posiziona su di essa, ed il programma entra automaticamente nel MODO CURSORE; se la ricerca non ha successo, viene invece stampato il messaggio "string not found", ed il puntatore conserva il suo valore iniziale.

X (FIND & SUBSTITUTE): Agisce come il comando precedente, con la differenza che, se la ricerca ha successo, la stringa trovata viene sostituita con la nuova stringa. Entrambi questi comandi sono ripetitivi, nel senso che se si vuole trovare due volte di seguito la stessa stringa, la seconda volta alla domanda "find?" è sufficiente rispondere con un return.

FRECCIA IN SU (CURSOR): Provoca l'ingresso nel MODO CURSORE, nel quale sono a disposizione vari comandi per spostare il cursore attraverso lo schermo, per inserire, cancellare, sostituire singoli caratteri, e per passare al modo OPERAZIONI CON I MARCHI.

Comandi nel modo cursore

FRECCIA A SINISTRA: Sposta il cursore a sinistra di una posizione, a meno che esso non sia già nella prima colonna; in tal caso il comando non viene eseguito.

FRECCIA A DESTRA: Sposta il

cursore a destra di una posizione, tranne che nel caso in cui il carattere attualmente puntato dal cursore sia l'ultimo della riga.

FRECCIA IN GIU': Sposta il cursore sul carattere più a sinistra della riga successiva, ammesso che questa esista. Se il cursore è già posizionato sull'ultima riga dello schermo, si ha uno "scroll" verso l'alto.

FRECCIA IN SU': Sposta il cursore sul carattere più a destra della riga precedente, ammesso che esista. Se il cursore è già sulla riga più alta dello schermo, si ha uno scroll verso il basso.

CHAR DEL: cancella dal video e dal testo il carattere puntato dal cursore, e copre il buco lasciato da esso spostando i caratteri successivi.

SHIFT + FRECCIA IN GIU': fa si che il prossimo carattere battuto venga inserito nella posizione attuale del cursore, e che i caratteri successivi vengano spostati per lasciargli il posto. Il cursore viene poi fatto avanzare di un posto (se già non è sull'ultimo carattere della riga).

CARATTERE QUALSIASI: viene sostituito a quello attualmente puntato dal cursore.

^: Causa l'ingresso nel modo OPERAZIONI CON I MARCHI. Da questo modo si può uscire o a seguito dell'esecuzione di una operazione con i marchi, o battendo il tasto E.

Comandi nel modo operazioni con i marchi

> (Setta marco iniziale): Il marco iniziale punta al carattere dove attualmente si trova il cursore. Ciò si nota facilmente osservando che quel carattere assume temporaneamente uno sfondo bianco.

< (Setta marco finale): il marco finale punta al carattere dove attualmente si trova il cursore. Il carattere in questione assume temporaneamente il colore giallo.

Z (Cancella i marchi): Resetta a zero il valore dei due marchi.

Seguito programma di editor.

```
1418 INPUT "MARGINE SINISTRO";MS%:PRINT
1420 PRINT CHR$(12);:GOSUB 10500
1430 IF F1%=1.0 THEN GOSUB 6000:F1%=0:GOTO 1510
1440 IF F%=1 THEN POKE #FF05,8:POKE #131,0
1450 COL%=0
1460 FOR I%=IPONT% TO FPONT%
1465 A=GETC:IF A=32.0 THEN GOSUB 10395
1470 IF COL%=N% THEN PRINT :COL%=0:GOSUB 1550
1480 PRINT CHR$(PEEK(I%));
1485 IF PEEK(I%)=13 THEN COL%=0:GOSUB 1550
1490 COL%=COL%+1
1500 NEXT I%
1510 POKE #131,1:POKE #FF05,#40
1515 PRINT
1520 GOSUB 10395:GOSUB 4000:GOTO 1995
1550 O%=PEEK(#131):POKE #131,1
1551 ROW%=ROW%-1:IF ROW%=0 THEN ROW%=1:CALLM #360:CURSOR 0,1:PRINT TAB(60);CURSOR 0,1
1554 POKE #131,O%:PRINT SPC(MS%);
1555 RETURN
1571 RETURN
1600 REM ----------------------
1605 REM LETTURA DI UN TESTO DAL NASTRO
1608 PRINT CHR$(12);:GOSUB 10500
1610 PRINT "RICORDI IL NOME DEL TESTO ? (S/N)"
1615 B=GETC:IF B=0.0 THEN 1615
1620 IF B=ASC("N") THEN 1630
1625 INPUT "NOME DEL TESTO";FILENAME$:PRINT
1630 PRINT "FAMMI LEGGERE"
1635 IF B=ASC("N") THEN LOADA TESTO% :GOTO 1650
1640 LOADA TESTO% FILENAME$
1650 PUNT%=PEEK(TEXBUF%-3)+256*PEEK(TEXBUF%-2)
1651 MAXPUNT%=PUNT%
1655 GOSUB 4000:GOTO 1995
1800 REM ---INSERIM. TESTO NUOVO NEL VECCHIO
1801 COLORT 14 0 0 0:GOSUB 10500
1802 RIT%=2
1803 GOSUB 10900:PRINT "MODO A";EXI$;"A";:CURSOR COL%,ROW%
1805 RISPUNT%=RISBUF%:GOSUB 10300
1810 A=GETC:IF A=0.0 THEN 1810
1812 IF A=22.0 THEN GOSUB 1300
1815 IF A<>ASC("#") THEN 1826
1820 GOSUB 10400
1825 IF A=ASC("A") THEN GOSUB 10000:GOSUB 10100:GOSUB 10200:GOTO 1995
1826 JPUNT%=MAXPUNT%+RISPUNT%-RISBUF%:GOTO 12000
1830 POKE RISPUNT%,A:RISPUNT%=RISPUNT%+1:COL%=COL%+1
1835 IF A=13 THEN PRINT CHR$(2);:COL%=0:ROW%=ROW%-1
1840 PRINT CHR$(A);
1845 IF A=8 THEN RISPUNT%=RISPUNT%-2:COL%=COL%-2
1850 IF COL%<0 THEN COL%=COL%+1:RISPUNT%=RISPUNT%+1:GOTO 1850
1855 IF COL%=MAXCOL% THEN PRINT :COL%=0:ROW%=ROW%-1
1860 IF ROW%<1 THEN ROW%=1:GOSUB 10600:CURSOR 0,1
1865 IF RISPUNT%=MAXBUF% THEN GOSUB 10000:GOSUB 10100:RISPUNT%=RISBUF%
1870 GOTO 1810
1990 REM --------------------
1991 REM MODULO DI COMANDO
1995 CURSOR 0,0:PRINT "ORDINA,PADRONE";TAB(59);:CURSOR COL%,ROW%
1996 COLORT 15 0 0 0
2000 GOSUB 10400
2001 COLORT 15 0 0 0:GOSUB 10500:GOSUB 10900
2010 IF A<>ASC("#") THEN RT%=0:GOTO 2030
2020 ON RT% GOTO 1030,3045
2021 GOTO 2000
2030 IF A=ASC("I") THEN 1000
2040 IF A=ASC("S") THEN 1200
2050 IF A=ASC("P") THEN IPONT%=TEXBUF%:FPONT%=MAXPUNT%:GOTO 1400
2060 IF A=ASC("L") THEN 1600
2070 IF A=ASC("A") THEN 1800
2080 IF A=16 THEN 3000
2090 IF A=ASC("T") THEN GOSUB 4000:GOTO 1995
2100 IF A=ASC("F") THEN 7000
2110 IF A=ASC("X") THEN 7300
2120 IF A=ASC("M") THEN GOSUB 2400:GOTO 2000
2199 GOTO 1995
2400 REM ---MOSTRA SPAZIO LIBERO--
2405 GOSUB 10900:CURSOR 0,0
2410 PRINT TEXTMAX%-MAXPUNT%;" BYTE LIBERI";:CURSOR COL%,ROW%:RETURN
3000 REM ------ MODO CURSORE -------
3001 COLORT 12 0 0 0:GOSUB 10500
3002 GOSUB 10900:PRINT "MODO CURSORE";EXI$;
3004 CURSOR COL%,ROW%
3005 GOSUB 10400:CURSOR COL%,ROW%
3007 COLORT 12 0 0 0:GOSUB 10500
3010 IF A=ASC("#") THEN RT%=2.0:GOTO 1995
3011 IF A=94.0 THEN 5000
3015 IF A<16.0 OR A>19.0 THEN 3025
3020 ON (A-15) GOSUB 3200,3300,3400,3500:GOTO 3005
3025 IF A=8 THEN GOSUB 3600:GOTO 3005
3030 IF A=21 THEN 3700
3045 COLORT 12 0 0 0:GOSUB 10500
3046 IF PEEK(PUNT%)=13 THEN FLAG%=1
3048 IF A=22.0 THEN GOSUB 1300
3050 POKE PUNT%,A:PRINT CHR$(A);:CURSOR COL%,ROW%
3052 IF FLAG%=1 THEN FLAG%=0:GOSUB 10300:GOSUB 10200
3055 GOTO 3005
3200 REM --- CURSORE IN SU ---
3250 PUNT%=PUNT%-(COL%+1)
3251 IF PUNT%<TEXBUF% THEN PUNT%=PUNT%+COL%+1:RETURN
3255 IF PEEK(PUNT%)=13 THEN 3275
3259 COL%=59
3260 ROW%=ROW%+1:IF ROW%=24 THEN GOSUB 15000
3261 CURSOR COL%,ROW%:RETURN
3275 POKE #575,(PUNT%-1) MOD 256:POKE #576,(PUNT%-1)/256:POKE #578,(TEXBUF%-1) MOD 256:POKE #579,(TEXBUF%-1)/256
3276 CALLM #570
3277 IPUNT%=PUNT%-PEEK(#59A)-PEEK(#59B)*256
3290 COL%=(IPUNT%+59) MOD 60:GOTO 3260
3300 REM ---- CURSORE IN GIU' ----
3301 IF ROW%>1 THEN 3324
3305 GOSUB 10600:CURSOR 0,1:PRINT TAB(59);:CURSOR COL%,2
3310 ROW%=2:GOSUB 10200:ROW%=2
3324 CNT%=MAXPUNT%-PUNT%:IF CNT%>255 THEN CNT%=255
3325 POKE #458,CNT%:POKE #45A,COL%:POKE #455,PUNT% MOD 256:POKE #456,PUNT%/256
3328 CALLM #450:ERR%=PEEK(#47F):IF ERR%=1 THEN RETURN
3330 PUNT%=PEEK(#455)+256*PEEK(#456):ROW%=ROW%-1:COL%=0:CURSOR COL%,ROW%:RETURN
3400 REM ---- CURSORE A SINISTRA ----
3401 IF COL%=0 THEN RETURN
3405 COL%=COL%-1:PUNT%=PUNT%-1:CURSOR COL%,ROW%:RETURN
3500 REM ---- CURSORE A DESTRA ----
3501 IF (COL%=59 OR PEEK(PUNT%)=13 OR PUNT%>=MAXPUNT%) THEN RETURN
3505 COL%=COL%+1:PUNT%=PUNT%+1:CURSOR COL%,ROW%:RETURN
3600 REM ---- DELETE ----
3602 INTRAS%=PUNT%+1:FINTRAS%=MAXPUNT%:INDES%=PUNT%:GOSUB 17000
3605 MAXPUNT%=MAXPUNT%-1
3609 GOSUB 10300
3610 COL%=COL%-1:IF COL%<0 THEN FLAG%=1:COL%=0:PUNT%=PUNT%+1
3611 CURSOR COL%,ROW%:PUNT%=PUNT%-1:GOSUB 10200:GOSUB 3500
3612 IF FLAG%=1 THEN FLAG%=0:GOSUB 3400
3615 RETURN
3700 REM ---- INSERT ----
3701 RIT%=3:JPUNT%=MAXPUNT%:GOTO 12000
3702 GOSUB 10400
3705 INTRAS%=MAXPUNT%:INDES%=MAXPUNT%+1:FINTRAS%=PUNT%:GOSUB 17100
3720 MAXPUNT%=MAXPUNT%+1
3722 IF A=13 THEN PRINT CHR$(32);:CURSOR COL%,ROW%
3723 IF A=22.0 THEN GOSUB 1300
3725 POKE PUNT%,A:GOSUB 10300:GOSUB 10200:GOSUB 3500:GOTO 3005
4000 REM ----TOP OF FILE-----
4001 PUNT%=TEXBUF%:CURSOR 0,23:COL%=0:ROW%=23:GOSUB 10300
4002 GOSUB 10200
4005 RETURN
5000 REM --- SPECIALI OPERAZIONI CON I MARCHI ---
5001 COLORT 8 15 0 0:GOSUB 10500
5005 GOSUB 10400
5010 IF A=94.0 THEN 3045
5020 IF A=ASC(">") THEN GOSUB 5100:GOTO 3005
5021 IF A=ASC("<") THEN GOSUB 5200:GOTO 3005
```

Un potente word processor

CHAR DEL (Cancella dal video e dal testo tutta la zona compresa tra i due marchi, estremi inclusi). Il buco che si crea viene coperto dallo spostamento all'indietro del testo successivo. Resetta anche il valore dei marchi.

I (INSERT): Inserisce, a partire dalla attuale posizione del cursore, la parte di testo compresa tra i due marchi. Il testo che prima era successivo al cursore viene spostato per lasciare posto all'inserimento.

S (SOVRAPPONI): La parte di testo compresa tra i due marchi viene sostituita al testo a partire dall'attuale posizione del cursore.

P (PRINT): Permette di stampare la parte di testo compresa tra i marchi.

NOTA BENE: I 4 precedenti comandi (CHAR DEL, INSERT, SOVRAPPONI e PRINT) hanno effetto solo se almeno uno dei marchi è stato settato. Se nessuno dei marchi è settato, i comandi non hanno effetto. Se solo uno dei marchi è settato, all'altro viene assegnato un valore di default; al marco iniziale viene assegnato l'inizio del buffer del testo (TEXBUF%), e al marco finale l'ultimo byte attualmente occupato nel buffer (MAXPUNT%).

Se invece entrambi i puntatori sono settati, ma il puntatore iniziale punta più avanti di quello finale, si considera solo l'ultimo puntatore settato come tale, e si assegna all'altro il valore di default.

E (EXIT): Consente di tornare al MODO CURSORE senza compiere alcuna operazione.

Alcune parole sulla stampa marginata

Come ho già detto prima, la stampa marginata fa uso delle possibilità grafiche della mia stampante (Centronics 154). In particolare, l'invio alla stampante della sequenza di caratteri "ESC % 0" mette la stampante in modo grafico, e l'invio successivo di un numero n di caratteri "SPACE" provoca un avanzamen-

Seguito programma di editor.

```
5022 IF A=8.0 THEN GOTO 5400
5023 IF A=ASC("Z") THEN GOSUB 5300:GOTO 3000
5024 IF A=ASC("I") THEN 5500
5025 IF A=ASC("S") THEN 5600
5026 IF A=ASC("P") THEN 5700
5098 IF A=ASC("E") THEN 3001
5099 GOTO 5005
5100 REM ---MARCO INIZIALE---
5101 GOSUB 15100:LASTMARK%=1:INIMARK%=PUNT%:COLORT 12 0 16-LASTMARK% 0:GOSUB 10500
5105 RETURN
5200 REM --- MARCO FINALE---
5201 GOSUB 15100:LASTMARK%=2:FINMARK%=PUNT%:COLORT 12 0 16-LASTMARK% 0
5202 GOSUB 10500
5205 RETURN
5300 REM ---CLEAR MARK---
5301 INIMARK%=0:FINMARK%=0:LASTMARK%=0:GOSUB 15200:RETURN
5400 REM --CANCELLA TRA I MARCHI--
5401 GOSUB 16000:IF ERR%=1.0 THEN ERR%=0:GOTO 3000
5405 MAXPUNT%=MAXPUNT%-FINMARK%+INIMARK%-1
5406 IF MAXPUNT%<=INIMARK% THEN MAXPUNT%=INIMARK%:GOTO 5430
5410 INTRAS%=FINMARK%+1:FINTRAS%=MAXPUNT%+FINMARK%-INIMARK%+1:INDES%=INIMARK%:GOSUB 17000
5420 GOSUB 5300
5430 GOSUB 4000
5440 GOTO 3000
5500 REM ---INSERISCI LA ZONA TRA I MARCHI---
5501 GOSUB 16000:IF ERR%=1.0 THEN ERR%=0:GOTO 3005
5502 RIT%=4:JPUNT%=MAXPUNT%+FINMARK%-INIMARK%+1:GOTO 12000
5510 INTRAS%=INIMARK%:FINTRAS%=FINMARK%:INDES%=RISBUF%:GOSUB 17000
5525 RISPUNT%=RISBUF%+FINMARK%-INIMARK%+1
5526 GOSUB 10000
5530 GOSUB 10100:GOSUB 4000:GOTO 3005
5600 REM ---SOVRAPPONI LA ZONA TRA I MARCHI---
5601 GOSUB 16000:IF ERR%=1 THEN ERR%=0:GOTO 3005
5602 RIT%=5:JPUNT%=PUNT%+FINMARK%-INIMARK%:GOTO 12000
5605 IF PUNT%+FINMARK%-INIMARK%>MAXPUNT% THEN MAXPUNT%=PUNT%+FINMARK%-INIMARK%
5610 INTRAS%=INIMARK%:FINTRAS%=FINMARK%:INDES%=PUNT%:GOSUB 17000
5620 GOSUB 4000:GOTO 3005
5700 REM ---  STAMPA TRA I MARCHI ---
5701 GOSUB 16000:IF ERR%=1 THEN ERR%=0:GOTO 3005
5702 IPONT%=INIMARK%:FPONT%=FINMARK%:GOSUB 1400
5705 GOTO 3000
6000 REM --STAMPA MARGINATA--
6004 GOSUB 6854
6005 WPUNT%=IPONT%
6010 IF PEEK(WPUNT%)=32 AND PEEK(WPUNT%+1)<>32 THEN WPUNT%=WPUNT%+1
6012 IF PEEK(WPUNT%)<>20 THEN 6015
6013 WPUNT%=WPUNT%+1:IF RIGA%>0 THEN GOSUB 6910
6015 IF WPUNT%>FPONT% THEN RETURN
6020 FOR I%=0 TO N%
6025 G%=PEEK(WPUNT%+I%):POKE WBUF%+I%,G%
6030 IF G%=13 OR WPUNT%+I%=FPONT% THEN GOSUB 6600:GOTO 6010
6035 NEXT I%
6040 IF G%=32.0 THEN GOSUB 6405:IF OFK%=1 THEN I%=N%:GOSUB 6600:GOTO 6010
6050 I%=N%
6055 G%=PEEK(WPUNT%+I%)
6060 POKE WBUF%+I%,G%
6065 IF G%=32 THEN GOSUB 6405:IF OFK%=1 THEN FSUP%=I%-1:GOTO 6100
6068 IF G%=13 THEN FSUP%=I%-1:FAD%=1:GOTO 6100
6070 IF PEEK(CHAR%+G%) IAND 1=1 THEN GOSUB 6410:IF OFK%=1 THEN FSUP%=I%:GOTO 6100
6075 IF I%=N%+10 THEN FSUP%=N%+10:GOTO 6100
6080 I%=I%+1:GOTO 6055
6100 I%=N%-1
6105 G%=PEEK(WBUF%+I%)
6110 IF G%=32 AND PEEK(WBUF%+I%-1)<>32 THEN GOSUB 6405:IF OFK%=1 THEN FINF%=I%-1:GOTO 6130
6115 IF PEEK(CHAR%+G%) IAND 1=1 THEN GOSUB 6410:IF OFK%=1 THEN FINF%=I%:GOTO 6130
6120 IF I%=N%-15 THEN FINF%=N%-15:GOTO 6130
6125 I%=I%-1:GOTO 6105
6130 GOSUB 6950
6170 SPAC2%=SPAC1%
6175 FOR I%=FINF%+1 TO FSUP%
6180 G%=PEEK(WBUF%+I%)
6185 IF G%=32 THEN SPAC2%=SPAC2%+1
6190 NEXT I%
6200 ADDOT=((N%-1.0)-FINF%)*7.0:AD=ADDOT/SPAC1%
6205 SUBDOT=(FSUP%-(N%-1.0))*7.0:SU=SUBDOT/SPAC2%
6210 IF AD>7.0 AND SU>2.0 THEN 6500
6220 IF AD<=2.0 THEN GOSUB 6700:GOTO 6250
6225 IF SU<=1.0 THEN GOSUB 6705:GOTO 6250
6230 IF AD<=4.0 THEN GOSUB 6700:GOTO 6250
6235 IF SU<=1.5 THEN GOSUB 6705:GOTO 6250
6240 IF AD<=7.0 THEN GOSUB 6700:GOTO 6250
6245 GOSUB 6705:GOTO 6250
6250 ETA%=(SPAC*7+DELTA)/SPAC:DIF%=SPAC*7+DELTA-ETA%*SPAC
6255 FOR I%=1 TO SPAC:DOD%(I%)=ETA%:NEXT I%
6265 IF DIF%=0 THEN 6275
6270 FOR I%=1 TO DIF%:DOD%(I%)=DOD%(I%)+1.0:NEXT I%
6275 J%=0
6280 FOR I%=WBUF% TO WBUF%+LIM%
6285 G%=PEEK(I%)
6290 IF G%=32.0 AND I%-WBUF%>=IACT% THEN GOSUB 6800:GOTO 6300
6295 POKE #FF06,G%:WAIT MEM #FF03,16,0
6300 NEXT I%
6301 GOSUB 6900:GOSUB 6850
6305 WPUNT%=WPUNT%+LIM%+1+FAD%:FAD%=0:GOTO 6010
6400 REM --- CONTROLLO ANDATA CAPO ---
6405 G1%=PEEK(WPUNT%+I%-1):G2%=PEEK(WPUNT%+I%+1):GOTO 6415
6410 G1%=G%:G2%=PEEK(WPUNT%+I%+1)
6415 OFK%=0
6420 IF (PEEK(CHAR%+G1%) IAND 32)+(PEEK(CHAR%+G2%) IAND 16)=0 THEN OFK%=1
6425 RETURN
6500 REM ---SILLABE--
6505 K%=N%-1
6510 IF K%<FINF% THEN 6750
6515 G%=PEEK(WBUF%+K%):IF PEEK(CHAR%+G%) IAND 128=0 THEN 6560
6520 G1%=PEEK(WBUF%+K%-1):G2%=PEEK(WBUF%+K%+1)
6525 IF G1%=G% THEN 6570
6530 IF (G1%=67 OR G1%=99) AND G%=G1%+14 THEN 6570
6535 IF (G%=83 OR G%=115) AND G2%<>G% THEN 6570
6540 IF PEEK(CHAR%+G1%) IAND 64=64 THEN 6570
6560 K%=K%-1:GOTO 6510
6570 POKE WBUF%+K%,45:FINF%=K%
6571 GOSUB 6950
6575 ADDOT=(N%-1.0-FINF%)*7.0:AD=ADDOT/SPAC1%
6576 WPUNT%=WPUNT%-1
6580 GOSUB 6700:GOTO 6250
6581 REM --------
6600 FOR J%=0 TO I%:WAIT MEM #FF03,16,0:POKE #FF06,PEEK(WBUF%+J%):NEXT J%
6605 WPUNT%=WPUNT%+I%+1
6610 GOSUB 6900
6615 GOSUB 6850:RETURN
6700 DELTA=ADDOT:LIM%=FINF%:SPAC=SPAC1%:RETURN
6705 DELTA=(-SUBDOT):LIM%=FSUP%:SPAC=SPAC2%:RETURN
6750 IF AD>9.0 THEN I%=N%-1:GOSUB 6600:GOTO 6010
6755 GOTO 6250
6800 J%=J%+1
6801 IF DOD%(J%)<7.0 THEN 6820
6805 POKE #FF06,32:WAIT MEM #FF03,16,0
6810 DOD%(J%)=DOD%(J%)-7.0
6815 GOTO 6801
6820 IF DOD%(J%)=0.0 THEN RETURN
6825 POKE #FF06,#1B:WAIT MEM #FF03,16,0:POKE #FF06,#25:WAIT MEM #FF03,16,0:POKE #FF06,#30:WAIT MEM #FF03,16,0
6830 FOR K%=1 TO DOD%(J%):POKE #FF06,32:WAIT MEM #FF03,16,0:NEXT K%
6835 POKE #FF06,#1B:WAIT MEM #FF03,16,0:POKE #FF06,14:WAIT MEM #FF03,16,0
6840 RETURN
6850 REM ---A CAPO e MARGINE SINISTRO---
6852 POKE #FF06,10:IF MS%=0 THEN RETURN
6854 FOR II%=1 TO MS%:POKE #FF06,32:WAIT MEM #FF03,16,0:NEXT II%
6856 RETURN
6900 REM ---SPAZIATURA---
6901 IF SPAZ%=0 THEN 6905
6902 FOR II%=1 TO SPAZ%:POKE #FF06,10:WAIT MEM #FF03,16,0:NEXT II%
6905 RIGA%=RIGA%+1:IF RIGA%<MAXRIG% THEN RETURN
6910 IF AUTOFORM%=1 THEN POKE #FF06,12
```

to orizzontale di n/7 di spazio da parte della testina. L'invio della sequenza "ESC SO" ripristina il normale funzionamento della stampante. Poiché non tutti hanno la stessa stampante, spiegherò cosa si deve fare per adattare il programma al proprio caso.

Per chi ha una stampante grafica: adattare le linee 6825-6835 alle caratteristiche della propria stampante (la linea 6825 invia la sequenza per mettere la stampante in modo grafico, la linea 6830 invia i caratteri di spazio, la linea 6835 riporta la stampante in modo non grafico.

Chi avesse infine intenzione di cambiare la logica con la quale si decide se allargare o stringere gli spazi di una riga, deve modificare le linee 6210-6240.

Per chi non ha una stampante grafica: sostituire le linee 6210-6305 con il listato 2.

Per evitare di andare a capo ad esempio dopo un apostrofo, o di cominciare una linea con una virgola, il programma svolge alcuni controlli basati sulle caratteristiche dei vari caratteri stampabili; queste caratteristiche sono codificate nella tabella CHAR%, tramite le linee DATA 30205-30210, nel modo illustrato in tabella 4.

Infine, per andare a capo ho usato il carattere di "LINE FEED" (codice ASCII = 10) nelle righe 6852 e 6902, si salta all'inizio della prossima pagina ogni volta che nel testo si trova il carattere SHIFT FRECCIA IN SU (codice ASCII 20) all'inizio di una riga, e, per quanto riguarda la velocità e il formato di trasmissione, l'istruzione POKE #FF05,8 (linea 111) fissa la velocità a 1200 baud, e il numero di bit di stop a 2.

Il controllo automatico sul "buffer pieno"

Per evitare che il testo oltrepassi la dimensione massima ammessa, e vada ad invadere zone della memoria destinate ad altri scopi, ogni vol-

Seguito programma di editor.

```
6915  IF AUTOFORM%=0 THEN GOSUB 10395
6920  RIGA%=0:RETURN
6950  REM ---CONTA SPAZI---
6951  SPAC1%=0:SPAC2%=0:ACT%=0
6952  IACT%=0
6955  FOR I%=0 TO FINF%
6960  G%=PEEK(WBUF%+I%)
6961  IF ACT%<>0 THEN 6964
6962  IF G%<>32 THEN ACT%=1:IACT%=I%
6963  GOTO 6965
6964  IF G%=32 THEN SPAC1%=SPAC1%+1
6965  NEXT I%
6970  RETURN
7000  REM ---FIND---
7001  GOSUB 10900:CURSOR 0,0:INPUT "Search for":ST1$
7002  IF LEN(ST1$)=0 THEN 7010
7004  L%=LEN(ST1$)
7006  FOR I%=0 TO L%-1:POKE SEARCHBUF%+I%,ASC(MID$(ST1$,I%,1)):NEXT I%
7010  IF L%=0 THEN 7001
7020  ISEARCH%=PUNT%
7021  CNT%=MAXPUNT%-(ISEARCH%+L%-1):IF CNT%<0 THEN 7025
7022  POKE #405,ISEARCH% MOD 256:POKE #406,ISEARCH%/256
7023  POKE #44A,L%:POKE #44B,CNT% MOD 256:POKE #44C,CNT%/256
7024  GOTO 7030
7025  CURSOR 0,0:PRINT "String not found":TAB(59)::CURSOR COL%,ROW%:GOTO 2000
7030  CALLM #400:ERR%=PEEK(#44D)
7035  IF ERR%=0 THEN 7025
7040  ISEARCH%=PEEK(#405)+256*PEEK(#406)
7045  PRINT CHR$(12)::GOSUB 10500:PUNT%=PUNT%-COL%:ROW%=23
7046  IF PUNT%=ISEARCH% THEN 7060
7047  CURSOR 0,0
7050  POKE #5A5,PUNT% MOD 256:POKE #5A6,PUNT%/256:POKE #5AB,0:POKE #5AD,ROW%
7051  POKE #608,(ISEARCH%) MOD 256:POKE #609,(ISEARCH%)/256
7052  INIZ%=#BFE7-#86*(23-ROW%)-2*(0):POKE #5A8,INIZ% MOD 256:POKE #5A9,INIZ%/256
7053  POINT%=#BFE7-#86*(24-ROW%):POKE #60A,POINT% MOD 256:POKE #60B,POINT%/256
7054  CALLM #5A0
7055  PUNT%=ISEARCH%
7060  COL%=PEEK(#60D):ROW%=PEEK(#60E)
7070  IF SOST%<>1 THEN 7075
7072  JPUNT%=MAXPUNT%+LSOS%-L%:GOTO 12000
7074  GOSUB 7500
7075  GOSUB 10200
7080  OC%=COL%:OR%=ROW%:GOSUB 3300
7090  IF PUNT%=ISEARCH% THEN 7100
7091  PUNT%=ISEARCH%:ROW%=OR%:COL%=OC%
7092  IF ROW%=1.0 THEN ROW%=2
7093  CURSOR COL%,ROW%
7100  GOTO 3000
7300  REM ---FIND AND SUBSTITUTE---
7301  GOSUB 10900:CURSOR 0,0:RIT%=6
7302  PRINT "Substitute"::INPUT S1$:PRINT " with"::INPUT S2$
7305  IF LEN(S1$)=0.0 THEN 7320
7307  L%=LEN(S1$)
7310  FOR I%=0 TO L%-1:POKE SEARCHBUF%+I%,ASC(MID$(S1$,I%,1)):NEXT I%
7320  IF L%=0 THEN 7301
7325  IF LEN(S2$)=0 THEN 7350
7330  LSOS%=LEN(S2$)
7335  FOR I%=0 TO LSOS%-1:POKE SOSBUF%+I%,ASC(MID$(S2$,I%,1)):NEXT I%
7350  SOST%=1
7360  GOTO 7020
7500  REM ---SUBROUTINE PER F&S---
7501  SOST%=0
7502  MAXPUNT%=MAXPUNT%-L%
7505  INTRAS%=ISEARCH%+L%:INDES%=ISEARCH%:FINTRAS%=MAXPUNT%+L%:GOSUB 17000
7510  IF LSOS%=0 THEN RETURN
7515  INTRAS%=MAXPUNT%:INDES%=MAXPUNT%+LSOS%:FINTRAS%=PUNT%:GOSUB 17100
7530  MAXPUNT%=MAXPUNT%+LSOS%
7535  FOR I%=0 TO LSOS%-1:POKE PUNT%+I%,PEEK(SOSBUF%+I%):NEXT I%
7540  RETURN
10000 REM ---ROUTINE SHIFT TEXT BUFFER
10001 IF RISPUNT%<=RISBUF% THEN RETURN
10005 INTRAS%=MAXPUNT%:INDES%=MAXPUNT%+RISPUNT%-RISBUF%:FINTRAS%=PUNT%:GOSUB 17100
10020 MAXPUNT%=MAXPUNT%+RISPUNT%-RISBUF%
10025 RETURN
10100 REM ----    INSERISCI NUOVO NEL VECCHIO
10101 IF RISPUNT%-1<RISBUF% THEN RETURN
10102 IF RISPUNT%-1=RISBUF% THEN POKE PUNT%,PEEK(RISBUF%):GOTO 10116
10105 INTRAS%=RISBUF%:FINTRAS%=RISPUNT%-1:INDES%=PUNT%:GOSUB 17000
10116 PUNT%=PUNT%+RISPUNT%-RISBUF%
10120 RETURN
10200 REM -----  STAMPA DA CURSORE A FINE SCHERMO
10201 CURSOR 0,0
10205 CNT%=MAXPUNT%-PUNT%:POKE #568,CNT% MOD 256:POKE #569,CNT%/256
10210 INIZ%=#BFE7-(23-ROW%)*#86-(COL%)*2:POKE #508,INIZ% MOD 256:POKE #509,INIZ%/256
10215 POINT%=#BFE7-#86*(24-ROW%):POKE #560,POINT% MOD 256:POKE #561,POINT%/256
10220 POKE #50B,COL%:POKE #50D,MAXCOL%:POKE #505,PUNT% MOD 256:POKE #506,PUNT%/256
10225 POKE #562,#DD:POKE #563,#B3
10230 CALLM #500
10245 CURSOR COL%,ROW%:RETURN
10300 REM ---CANCELLA DA CURSORE A FINE SCHERMO
10305 IF COL%<59 THEN CURSOR COL%+1,ROW%:PRINT TAB(60);
10310 IF ROW%=1 THEN 10330
10315 INIZ%=#BFE7-(23-ROW%)*#86-(COL%+1)*2
10316 POKE #38F,ROW%
10318 POKE #3C5,INIZ% MOD 256:POKE #3C6,INIZ%/256
10320 POKE #3C8,62-COL%:POKE #39F,32
10322 CALLM #3C0
10324 POKE #39F,0
10330 CURSOR COL%,ROW%:RETURN
10395 Y%=CURY:CURSOR 0,0:PRINT "BATTI UN CARATTERE PER CONTINUARE";:CURSOR COL%,Y%
10400 REM ---LEGGI UN CARATTERE---
10405 POKE #2BA,0:POKE #2BB,0:POKE #2BC,0:POKE #2BD,0
10410 A=GETC:IF A=0.0 THEN 10410
10415 RETURN
10500 REM ---SET COLORE ULTIMA RIGA---
10501 POKE #B3E4,#CB
10502 RETURN
10600 REM --SCROLL ALTO—
10605 CALLM #360:CURSOR 0,1:PRINT TAB(60)::CURSOR 0,1
10610 RETURN
10700 REM --SCROLL VERSO BASSO--
10705 CALLM #3D0
10706 CURSOR CURX,22:PRINT CHR$(T%);
10708 CURSOR 0,23:PRINT TAB(60)::CURSOR 0,23
10710 RETURN
10800 REM ---STAMPA UN CARATTERE--
10801 G%=PEEK(I%):IF G%=13 THEN PRINT CHR$(2);
10802 PRINT CHR$(G%)::RETURN
10900 REM -CANCELLA RIGA BASSA---
10901 CURSOR 0,0:PRINT TAB(60)::CURSOR 0,0:RETURN
12000 REM ---CONTROLLO SU BUFFER PIENO---
12001 IF JPUNT%+60>TEXTMAX% THEN 12020
12010 ON RIT% GOTO 1031,1830,3702,5510,5605,7074
12020 GOSUB 10900
12025 IF JPUNT%>=TEXTMAX% THEN 12040
12030 PRINT "ATTENZIONE:SOLO ";TEXTMAX%-JPUNT%;" BYTE LIBERI";
12035 CURSOR COL%,ROW%:GOTO 12010
12040 PRINT "BUFFER PIENO";:CURSOR COL%,ROW%
12045 ON RIT% GOTO 2000,2000,3005,3005,3005,3005
15000 REM --SCROLL IN SU---
15001 T%=PEEK(PUNT%+CURX+1):IF T%=13 THEN T%=2
15002 PUNT%=PUNT%-COL%:GOSUB 10700:ROW%=23:CURSOR 0,23
15004 IF COL%=0 THEN 15011
15005 FOR I%=PUNT% TO PUNT%+COL%-1
15006 PRINT CHR$(PEEK(I%))::NEXT I%
15010 PUNT%=PUNT%+COL%:IF CURX=59 THEN PRINT CHR$(PEEK(PUNT%))::RETURN
15011 IF PEEK(PUNT%)=13 THEN PRINT CHR$(2);
15012 RETURN
15100 POKE #BFE4-(COL%*2)-(23-ROW%)*#86,55
15101 RETURN
15200 REM ---RESET COLORI SCHERMO---
15205 CALLM #390
15210 RETURN
16000 REM --ROUTINE DI PROTEZIONE--
```

ta che si compie una operazione che incrementa il numero di caratteri del testo viene anche eseguita una subroutine che controlla se c'è ancora spazio a disposizione. Ogni operazione che abbia come risultato il superamento del limite imposto al buffer del testo viene impedita, e ogni operazione che lasci liberi meno di 60 byte in questo buffer viene accompagnata da una scritta diagnostica di avvertimento sulla riga bassa dello schermo.

Cosa fare in caso di "incidente"

Come ho già detto all'inizio, il fatto che la maggior parte del programma sia scritta in BASIC permette di trarsi di impaccio in alcuni casi che altrimenti potrebbero avere spiacevoli conseguenze. Un comando sbagliato, dato per la stanchezza quando siete a tre righe dalla fine del vostro testo, un tasto premuto erroneamente, un eventuale "baco" del programma non vi lascerà ad imprecare contro la malasorte cercando nel frattempo il coltello da harakiri, con il quale porre dignitosamente fine alla vostra esistenza ormai senza alcun senso, ma, nella maggior parte dei casi, vi consentirà ugualmente di salvare il vostro prezioso lavoro. Facciamo alcuni esempi:

Avete dato un comando INSERT al posto di ADD: per salvare la situazione dovete:

— interrompere il programma con un "BREAK",

— scrivere MAXPUNT% = TEXBUF% + numero caratteri che pensate formino il vostro testo,

— riprendere il programma con un "CONT",

— battere un "#" per tornare in MODO COMANDO,

— eventualmente mettersi in MODO CURSORE per controllare se il valore di MAXPUNT% dato era corretto (e se non lo era tornate in MODO COMANDO e riiniziate la sequenza da capo),

— possibilmente salvare il testo su

Seguito programma di editor.

```
16001 IF INIMARK%+FINMARK%=0 THEN ERR%=1:RETURN
16002 IF INIMARK%=0 THEN INIMARK%=TEXBUF%
16003 IF FINMARK%=0 THEN FINMARK%=MAXPUNT%
16005 IF INIMARK%<FINMARK% THEN 16010
16006 IF LASTMARK%=1 THEN FINMARK%=MAXPUNT%:GOTO 16010
16008 INIMARK%=TEXBUF%
16010 RETURN
17000 REM ----TRASF1----
17005 POKE #305,INTRAS% MOD 256:POKE #306,INTRAS%/256
17010 POKE #308,INDES% MOD 256:POKE #309,INDES%/256
17015 POKE #30B,FINTRAS% MOD 256:POKE #30C,FINTRAS%/256
17020 CALLM #300
17025 RETURN
17100 REM ----TRASF2----
17105 POKE #325,INTRAS% MOD 256:POKE #326,INTRAS%/256
17110 POKE #328,INDES% MOD 256:POKE #329,INDES%/256
17115 POKE #32B,FINTRAS% MOD 256:POKE #32C,FINTRAS%/256
17120 CALLM #320
17125 RETURN
20000 CURSOR 40,1:PRINT PUNT%;:CURSOR COL%,ROW%:RETURN
30000 REM --ROUTINE IN ASSEMBLER---
30001 FOR I%=#300 TO #47F:READ A%:POKE I%,A%:NEXT I%
30004 REM ---TRASFERIMENTI---
30005 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,0,0,1,0,0,#CD,#40,3
30010 DATA #E1,#D1,#C1,#F1,#C9,0,0,0,0,0,0,0,0,0,0,0
30015 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,0,0,1,0,0,#CD,#50,3
30020 DATA #E1,#D1,#C1,#F1,#C9,0,0,0,0,0,0,0,0,0,0,0
30025 DATA #7E,#12,#7C,#B8,#C2,#4A,3,#7D,#B9,#C8,#23,#13,#C3,#40,3,0
30030 DATA #7E,#12,#7C,#B8,#C2,#5A,3,#7D,#B9,#C8,#2B,#1B,#C3,#50,3,0
30034 REM ---SCROLL VERSO IL BASSO---
30035 DATA #F5,#C5,#D5,#E5,#21,#61,#BF,#11,#E7,#BF,1,#ED,#B3,#7E,#12,#7C
30040 DATA #B8,#C2,#79,3,#7D,#B9,#CA,#80,3,#2B,#2B,#1B,#1B,#C3,#6D,3
30045 DATA #E1,#D1,#C1,#F1,#C9,0,0,0,0,0,0,0,0,0,0,0
30054 REM ---RESET COLORI SCHERMO---
30055 DATA #F5,#C5,#D5,#E5,#21,#EC,#BF,#3E,23,#32,#BF,3,#3E,66,#36,0
30060 DATA #3D,#CA,#A9,3,#2B,#2B,#C3,#9E,3,#3A,#BF,3,#3D,#CA,#BA,3
30065 DATA #32,#BF,3,#2B,#2B,#2B,#2B,#C3,#9C,3,#E1,#D1,#C1,#F1,#C9,0
30069 REM ---CANCELLA DA CURSORE IN GIU'---
30070 DATA #F5,#C5,#D5,#E5,#21,0,0,#3E,0,#C3,#9E,3,0,0,0,0
30074 REM ---SCROLL VERSO L'ALTO---
30075 DATA #F5,#C5,#D5,#E5,#21,#73,#B4,#11,#ED,#B3,1,#E7,#BF,#7E,#12,#7C
30080 DATA #B8,#C2,#E9,3,#7D,#B9,#CA,#F0,3,#23,#23,#13,#13,#C3,#DD,3
30085 DATA #E1,#D1,#C1,#F1,#C9,0,0,0,0,0,0,0,0,0,0,0
30089 REM ---SEARCH---
30090 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,#CE,#63,#3A,#4A,4,#47,#7E,#EB
30095 DATA #96,#EB,#CA,#2C,4,#23,#22,5,4,#2A,#4B,4,#7C,#B7,#C2,#3F
30100 DATA 4,#7D,#B7,#C2,#3F,4,#32,#4D,4,#C3,#35,4,5,#C2,#3A,4
30105 DATA #3E,1,#32,#4D,4,#E1,#D1,#C1,#F1,#C9,#23,#13,#C3,#E,4,#2B
30110 DATA #22,#4B,4,#C3,4,4,0,0,0,0,0,0,0,0,0,0
30114 REM ---CURSURE GIU'---
30115 DATA #F5,#C5,#D5,#E5,#21,0,0,6,0,#E,6,#23,5,#C2,#6A,4
30120 DATA #3E,1,#32,#7F,4,#E1,#D1,#C1,#F1,#C9,#2B,#7E,#23,#D6,13,#CA
30125 DATA #79,4,#C,#79,#D6,60,#C2,#5B,4,#22,#55,4,#C3,#62,4,0
30200 REM --- DATI CHAR---
30201 CHAR%=#460
30202 FOR I%=CHAR%+32 TO CHAR%+127:READ C%:POKE I%,C%:NEXT I%
30205 DATA 0,21,4,2,2,18,50,52,36,21,0,17,21,1,21,2
30206 DATA 2,4,2,2,2,2,2,2,2,2,21,21,2,2,2,17
30207 DATA 2,2,130,130,130,2,130,130,130,4,132,130,194,194,194,2
30208 DATA 130,130,194,130,130,2,130,130,130,130,130,37,3,21,4,0
30209 DATA 21,0,128,128,128,0,128,128,128,4,132,128,192,194,192,0
30210 DATA 128,128,192,128,128,0,128,128,128,128,128,37,5,21,0,0
30300 FOR I%=#500 TO #62F:READ A%:POKE I%,A%:NEXT I%
30301 REM --- ROUTINE 10200--
30305 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,0,0,6,0,#E,0,#7E,#FE
30306 DATA 13,#CA,#6A,5,#EB,#77,#2B,#2B,#EB,4,#78,#B9,#CA,#38,5,#E5
30307 DATA #2A,#68,5,#2B,#22,#68,5,#7C,#B7,#C2,#2E,5,#7D,#B7,#E1,#CA
30308 DATA #5A,5,#23,#C3,#E,5,#36,2,#EB,#2A,#60,5,#EB,#E5,#2A,#62
30309 DATA 5,#7A,#BC,#C2,#4B,5,#7B,#BD,#CA,#59,5,#26,#FF,#2E,#7A,#19
30310 DATA #22,#60,5,#E1,6,0,#C3,#1F,5,#E1,#E1,#D1,#C1,#F1,#C9,0
30311 DATA 0,0,0,0,0,0,0,0,0,0,#EB,#36,2,#C3,#39,5
30330 REM ---CURSORE SU---
30331 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,0,0,#7E,#FE,13,#CA,#8E,5,#7D
30332 DATA #BB,#C2,#8A,5,#7C,#BA,#CA,#8E,5,#2B,#C3,#7A,5,#22,#9A,5
30333 DATA #E1,#D1,#C1,#F1,#C9,0,0,0,0,0,0,0,0,0,0,0
30350 REM ---PRINT FIND---
30351 DATA #F5,#C5,#D5,#E5,#21,0,0,#11,0,0,6,0,#E,0,#7E,#FE
30352 DATA 13,#CA,#D8,5,#EB,#77,#2B,#2B,4,#78,#D6,60,#CA,#DB,5,#13
30353 DATA #E5,#2A,8,6,#7A,#BC,#C2,#CB,5,#7B,#BD,#E1,#CA,#D3,5,#EB
30354 DATA #C3,#AE,5,#C3,#23,6,0,0,#EB,#AF,#77,#47,#2A,#A,6,#D
30355 DATA #CA,#F0,5,#EB,#E5,#21,#7A,#FF,#19,#22,#A,6,#E1,#C3,#0,6
30356 DATA #CD,#60,3,#CD,#10,6,#E,1,#EB,#E5,#21,#86,0,#19,#EB,#E1
30357 DATA #EB,#C3,#BF,5,0,0,0,0,0,0,0,0,0,0,0,0
30358 DATA #E5,#C5,#21,#63,#B4,6,60,#36,32,#2B,#2B,5,#C2,#17,6,#C1
30359 DATA #E1,#C9,0,#78,#32,#D,6,#79,#32,#E,6,#E1,#D1,#C1,#F1,#C9
30500 RETURN
```

```
6210  IF AD>7.0 THEN 6500
6245  GOSUB 6700:GOTO 6250
6250  DIF%=ADDOT
6255  FOR I%=1 TO SPAC:DOD%(I%)=7:NEXT I%
6265  IF DIF%=0 THEN 6275
6270  FOR I%=1 TO SPAC:DOD%(I%)=DOD%(I%)+7.0
6271  ADDOT=ADDOT-7.0:IF ADDOT=0.0 THEN GOSUB 6390:GOTO 6275
6272  NEXT I%
6273  GOTO 6270
6275  J%=0
6280  FOR I%=WBUF% TO WBUF%+LIM%
6285  G%=PEEK(I%)
6290  IF G%=32.0 AND I%-WBUF%>=IACT% THEN GOSUB 6800:GOTO 6300
6295  POKE #FF06,G%:WAIT MEM #FF03,16,0
6300  NEXT I%
6301  GOSUB 6900:GOSUB 6850
6305  WPUNT%=WPUNT%+LIM%+1+FAD%:FAD%=0:GOTO 6010
6390  FOR H%=1 TO 10:H1%=RND(0.0)*(SPAC-1.0):H2%=RND(0.0)*(SPAC-1.0)
6391  H3%=DOD%(H1%+1.0):DOD%(H1%+1.0)=DOD%(H2%+1.0):DOD%(H2%+1.0)=H3%
6392  NEXT H%
6393  RETURN
```

Listato 2. *Chi non utilizza una stampante grafica, sostituisca questa linea alla corrispondente del listato 1.*

cassetta.

Avete inavvertitamente premuto il tasto BREAK e il programma si è interrotto: a seconda che il comando CONT venga accettato o no, si hanno due casi:

1) CONT è accettato: potete continuare normalmente, a meno che il programma non fosse in MODO CURSORE. In questo caso, poiché si è persa la corrispondenza tra la posizione del cursore sullo schermo e il carattere puntato dal puntatore nel buffer del testo, vi conviene uscire dal modo cursore, dare eventualmente un comando "T", e poi riprendere il vostro lavoro.

2) CONT non è accettato (ad esempio se stava girando una routine Assembler); allora vi conviene salvare il testo facendo ripartire il programma dalla linea 1200 (RUN 1200); se il programma si dovesse interrompere per conto suo, ad esempio per STACK OVERFLOW ON LINE 1220, fatelo ancora ripartire dalla linea 1235. Dopodiché la cosa migliore è fare ripartire il programma dall'inizio (RUN), e leggere il testo appena salvato.

Se quanto detto avviene quando siete in MODO A, potete perdere in questo modo i caratteri che non sono ancora stati trasferiti dal buffer RISBUF% al buffer del testo (al massimo 500 caratteri).

NOTA: la pressione simultanea dei tre tasti REPT, CHAR DEL e SHIFT ha lo stesso effetto di un BREAK.

Avvertenze per chi volesse modificare il programma

Questo programma è stato progettato, codificato, perfezionato e corretto in pochi giorni; per forza di cose c'è ancora molto spazio per miglioramenti. Piccole modifiche, inserimento di nuovi comandi, ed eventuali correzioni, non dovrebbero costare molta fatica. Occorre però tenere presente che:

— una delle routine Assembler fa uso del valore di SEARCHBUF%; se si modifica questo valore occorre modificare anche la linea 30090 sostituendo a #CE, #63 il nuovo valore,

— alcuni indirizzi e parametri vengono passati alle routine Assembler in modo un po' sporco, sostituendoli nel codice; occorre perciò fare attenzione a non modificare questo passaggio di parametri, per evitare inconvenienti. ■

Ziup, zip e swoop

Suono, non parole per Apple II

di *Alessandro Stecchina*

Chi di noi, entrando in una arcade (sala di videogiochi) e sentendo il frastornante e incessante suono di centinaia di laser, missili, bombe ed esplosioni, non ha desiderato essere in grado di riprodurli sul proprio Apple per inserirli nei propri giochi o nei propri programmi più divertenti?

Certamente molti avranno provato ad ottenere risultati simili in BASIC, ma con scarsi risultati.

Infatti il BASIC non permette quella velocità necessaria per ottenere i suoni molto acuti e gli effetti speciali richiesti per conseguire il risultato desiderato.

In questo articolo vedremo alcune routine che servono a generare suoni adatti a giochi e programmi vari.

Queste routine potrete usarle in linguaggio macchina o insieme ad un programma BASIC che le gestisca.

Come fare un suono controllabile

Il listato 1 mostra il sorgente e il codice oggetto di una semplice routine che genera un suono.

La frequenza e la durata del suono sono controllati mettendo questi valori nella locazione di memoria 301 e 303, per esempio usando il programma BASIC del listato 2.

Il contenuto della locazione di memoria 301 è direttamente proporzionale al periodo della nota, quindi

```
FILE NAME      :  UNA NOTA
CURRENT DATE:  28/11/83

:ASM

                1000 *-----------------------------
                1010 *        UNA NOTA
                1020 *-----------------------------
C030-           1030 ALTOP   .EQ $C030
                1040 *-----------------------------
                1050         .OR $300
0300- A0 00     1060 SUONO   LDY #0         CARICA DURATA
0302- A2 00     1070 .1      LDX #0         CARICA PERIODO
0304- AD 30 C0  1080         LDA ALTOP      FAI CLICK
0307- CA        1090 .2      DEX            LOOP DEL PERIODO
0308- D0 FD     1100         BNE .2
030A- 88        1110         DEY            LOOP DI DURATA
030B- D0 F5     1120         BNE .1
030D- 60        1130         RTS

SYMBOL TABLE

C030- ALTOP
0300- SUONO
.01=0302, .02=0307

0000 ERRORS IN ASSEMBLY
:
```

Listato 1. *Routine per la generazione di un suono.*

```
10  INPUT "QUALE E' LA DURATA DEL
     LA NOTA  ? (0-255) ";D
20  POKE 769,D
30  INPUT "QUALE E' IL PERIODO DE
     LLA NOTA ? (0-255) ";P
40  POKE 771,P
50  CALL 768
```

Listato 2. *Programma per caricare i valori utilizzati dalla routine del listato 1.*

Ziup, zip e swoop

```
FILE NAME    :  RAFFICA
CURRENT DATE:  28/11/83

:ASM

                  1000 *------------------------------------
                  1010 *         RAFFICA
                  1020 *------------------------------------
C030-             1030 ALTOP   .EQ $C030
0000-             1035 CNTR    .EQ $00
                  1037 *------------------------------------
                  1050         .OR $300
0300- A2 40       1060 START   LDX #64      LUNGHEZZA DEL COLPO
0302- A9 0A       1070         LDA #10      NUMERO DEI COLPI
0304- 85 00       1072         STA CNTR
0306- AD 30 C0    1080 .2      LDA ALTOP    FAI CLICK
0309- BC 00 BA    1082         LDY $BA00,X  DURATA IMPULSO RANDOM
030C- 88          1085 .1      DEY          LOOP DI DURATA PER IL SINGOLO SUONO
030D- D0 FD       1087         BNE .1
030F- CA          1090         DEX          LOOP DEL PERIODO
0310- D0 F4       1100         BNE .2
0312- C6 00       1110         DEC CNTR     PRENDI IL PROSSIMO COLPO
0314- D0 F0       1120         BNE .2
0316- 60          1130         RTS

SYMBOL TABLE

C030- ALTOP
0000- CNTR
0300- START
.02=0306, .01=030C

0000 ERRORS IN ASSEMBLY
:PR
```

Listato 3. *Questa subroutine genera una raffica di "raggi protonici".*

```
                  1000 *------------------------------------
                  1010 *     LASER "SWOOP"
                  1020 *------------------------------------
C030-             1030 ALTOP   .EQ $C030
0000-             1040 CONT.IMP .EQ $00
0001-             1050 LARG.IMP .EQ $01
0002-             1060 CONT.SWOOP .EQ $02
                  1070 *------------------------------------
                  1075         .OR $300
0300- A9 01       1080 SWOOP   LDA #1       UN IMPULSO PER CIASCUNA LUNGHEZZA
0302- 85 00       1090         STA CONT.IMP
0304- A9 A0       1100         LDA #160     COMINCIA CON LA MASSIMA LARGHEZZA
0306- 85 01       1110         STA LARG.IMP
0308- A4 00       1120 .1      LDY CONT.IMP
030A- AD 30 C0    1130 .2      LDA ALTOP
030D- A6 01       1140         LDX LARG.IMP
030F- CA          1150 .3      DEX          LOOP PER UN IMPULSO
0310- D0 FD       1160         BNE .3
0312- 88          1170         DEY          LOOP PER IL NUMERO DI IMPULSI
0313- D0 F5       1180         BNE .2       A CIASCUNA LARGHEZZA DI IMPULSO
0315- C6 01       1190         DEC LARG.IMP RIDUCI LARGHEZZA
0317- D0 EF       1200         BNE .1
0319- 60          1210         RTS
                  1220 *------------------------------------
                  1230 *  RAFFICA DI SWOOPS
                  1240 *------------------------------------
031A- A9 0A       1250 SWOOP2  LDA #10      NUMERO DI COLPI
031C- 85 02       1260         STA CONT.SWOOP
031E- 20 00 03    1270 .1      JSR SWOOP
0321- C6 02       1280         DEC CONT.SWOOP
0323- D0 F9       1290         BNE .1
0325- 60          1300         RTS

SYMBOL TABLE

C030- ALTOP
0000- CONT.IMP
0002- CONT.SWOOP
0001- LARG.IMP
0300- SWOOP
.01=0308, .02=030A, .03=030F
031A- SWOOP2
.01=031E

0000 ERRORS IN ASSEMBLY
:P
```

Listato 4. *Routine Laser. Potrete utilizzarla per i vostri giochi spaziali.*

è inversamente proporzionale alla frequenza della stessa.

La locazionedi memoria 303 invece contiene il numero di volte per cui deve ripetersi il "click" dell'altoparlante, cioè è proporzionale alla durata della nota.

Raffica

Ma se noi non vogliamo suonare la ninna-nanna, ma dobbiamo ingaggiare una lotta senza quartiere con degli invasori alieni discesi con un disco (volante, non floppy)?

Sostanzialmente l'intervallo di tempo tra un click e l'altro è generato in maniera pseudo-casuale, pescandolo da una zona di memoria del computer. Si deve scegliere una zona che ovviamente contenga dei dati, pertanto si è scelta una zona del DOS.

Ecco il programma del listato 3 che genera una bella raffica di raggi protonici.

Nella locazione di memoria $303 è contenuto il numero di colpi della raffica, mentre nella locazione di memoria $301 c'è la lunghezza di ciascun colpo.

Laser, l'arma che salverà la Terra

Non poteva mancare infine, e la trovate nel listato 4, una routine per il laser, l'arma delle guerre spaziali per antonomasia.

La routine che parte alla locazione di memoria $300 fa un solo "swoop", quella che invece inizia alla locazione di memoria $31A permette di fare una raffica di laser (con certi alieni è necessaria).

Alla locazione di memoria $31B si può inserire il numero di "swoop" desiderati.

Se con nessuna delle routine presentate in questo numero riuscite ad abbattere l'alieno che sta sul video, allora credo che sia meglio che giochiate a Lemonade. Dateci dentro!! ■

Semplice renumber per Spectrum

Questo renumber è molto efficiente per quanto riguarda i numeri di linea, ma trascura i GOTO, i GOSUB ed i RESTORE

a cura della *Redazione*

La routine in linguaggio macchina presentata ci è stata inviata da Luciano Andrei di Lastra a Signa (FI). Funziona così: all'inizio HL punta il primo byte del programma BASIC, cioè il byte alto del numero di linea, mentre DE contiene il numero di linea di partenza (i due byte che contengono il numero di linea sono memorizzati nell'ordine: byte più significativo, byte meno significativo; cfr. p. 218 del manuale italiano). In BC viene caricata la lunghezza della linea e dopo ADD HL, BC, HL punta il primo byte dopo la fine della linea.

Siccome i numeri di linea arrivano fino a 9999, il loro byte alto (o più significativo) è sicuramente inferiore a 40, mentre se è iniziata la zona variabili, si ha un valore maggiore od uguale a 64. Le istruzioni CP 64 e RET NC causano l'abbandono della routine se si è arrivati in tale zona, in caso contrario a DE viene sommato il "passo" e si ritorna alla terza istruzione.

Accanto alla routine viene presentato un programma LD & CK che carica automaticamente la routine subito sotto RAMTOP, e gira quindi senza modifiche sia sul 16 Kbyte che sul 48 Kbyte. Questo programma incorpora una routine che stampa su ZX Printer quello che ha appena caricato, sia in codice decimale che in codice esadecimale. Per la stampa su video cambiare tutti gli LPRINT in PRINT.

La routine così caricata può essere memorizzata con:
SAVE "renumcode" CODE ramtop + 1,28
per il registratore a cassette, e con:
SAVE ★ "m"; 1; "renumcode" CODE ramtop + 1,28
per il Microdrive.

Prima di poter impiegare la routine occorre inserire il primo numero di linea ed il passo desiderati. Questa funzione viene svolta dal programma renumber presentato. ◀

```
65340, 28
INDIR  DEC   HEX
65340  42    2A
65341  83    53
65342  92    5C
65343  17    11
65344  0     00
65345  0     00
65346  114   72
65347  35    23
65348  115   73
65349  35    23
65350  78    4E
65351  35    23
65352  70    46
65353  35    23
65354  9     09
65355  126   7E
65356  254   FE
65357  64    40
65358  208   D0
65359  229   E5
65360  33    21
65361  0     00
65362  0     00
65363  25    19
65364  235   EB
65365  225   E1
65366  24    18
65367  234   EA
```

Figura 1. *Output del programma LD & CK. Usatelo per controllare l'esattezza dei vostri codici.*

LD HL, (23635)	42, 83, 92
LD DE, partenza	17, 0, 0
LD (HL), D	114
INC HL	35
LD (HL), E	115
INC HL	35
LD C, (HL)	78
INC HL	35
LD B, (HL)	70
INC HL	35
ADD HL, BC	9
LD A, (HL)	126
CP 64	254, 64
RET NC	208
PUSH HL	229
LD HL, passo	33, 0, 0
ADD HL, DE	25
EX DE, HL	235
POP HL	225
JR - 22	24, 234

Listato 1. *Assembler della routine che effettua la numerazione delle linee.*



Semplice renumber per Spectrum

```
   1 REM LDeCK
   2 REM RENUMBER
   5 LET ramtop=PEEK 23730+256*P
EEK 23731
  10 CLEAR ramtop-28: REM  RAMTO
P-n
  15 LET ramtop=PEEK 23730+256*P
EEK 23731
  30 LET a=ramtop+1: LET n=28
  40 GO SUB 210
  50 GO SUB 130
  60 STOP
 130 LPRINT a;", ";n
 135 LPRINT "INDIR";TAB 7;"DEC";
TAB 12;"HEX"
 140 FOR i=0 TO n-1
 150 LET d=PEEK (a+i)
 155 LET n=d
 160 GO SUB 1000
 170 LPRINT a+i;TAB 7;n;TAB 12;b
$
 180 NEXT i
 190 LPRINT
 200 RETURN
 210 FOR i=0 TO n-1
 220 READ d
 230 POKE a+i,d
 240 NEXT i
 250 RETURN
 260 REM dati
 300 DATA 42,83,92,17,0,0,114,35
,115,35,78,35,70,35,9,126,254,64
,208,229,33,0,0,25,235,225,24,23
4
1000 LET c$="0123456789ABCDEF"
1010 LET b$=""
1020 FOR k=1 TO 0 STEP -1
1030 LET b=INT (d/16↑k)
1040 LET b$=b$+c$(b+1)
1050 LET d=d-b*16↑k
1060 NEXT k
1070 RETURN
```

Listato 2. *Un programma sempre utile per caricare i codici. La variabile n alla linea 30 contiene il numero di byte da caricare. I dati sono alla linea 300. Viene sfruttato automaticamente lo spazio più vicino alla RAMTOP. Il programma gira sia per la versione 16 Kbyte che 48 Kbyte.*

Siccome occupa i numeri di linea alti può essere caricato con MERGE e poi caricare la routine rispondendo con un qualsiasi carattere ed ENTER alla prima domanda. Il programma prevede l'uso del Microdrive; se si usa invece il registratore a cassette occorre modificare la linea 9080 così:

9080 LOAD "renumcode" CODE begin, 28

ed ovviamente, predisporre la cassetta. Per usi consecutivi rispondere solo con ENTER alla prima domanda.

Osservate infine cosa succede se i nuovi numeri di linea superano 9999.

Vi suggeriamo un paio di modifiche che potreste tentare ed inviarci mi raccomando su cassetta) se riesce bene (naturalmente occore usare il linguaggio macchina).

1) Rinumerare le linee a partire da una certa linea del programma originale e fino ad una certa altra linea, non incondizionatamente dall'inizio alla fine.
2) Aggiornare i GOTO, i GOSUB, i RESTORE in conseguenza alla nuova numerazione.
3) Qualunque estensione che possa essere utile.

Buona fortuna! ■

```
9000 REM RENUMBER con GOTO 9000
9010 LET begin=1+PEEK 23730+256*
PEEK 23731
9020 INPUT "PREMERE ENTER SE IL
CODICE E'   GIA' STATO CARICATO
 ";d$: IF d$<>"" THEN GO SUB 908
0
9030 INPUT "PRIMA LINEA = ";p'"P
ASSO = ";a
9040 POKE begin+4,p-INT (p/256)*
256: POKE begin+5,INT (p/256)
9050 POKE begin+21,a-INT (a/256)
*256: POKE begin+22,INT (a/256)
9060 RANDOMIZE USR begin
9070 STOP
9080 LOAD *"m";1;"renumcode"CODE
 begin,28
9090 RETURN
```

Listato 3. *Il programma permette di impiegare con facilità la routine di numerazione; da usare con GOTO 9000.*

Ingrandimento e riduzione di caratteri

Modificate a vostro piacere la dimensione dei caratteri del vostro ZX Spectrum

di *Marcello Spero*

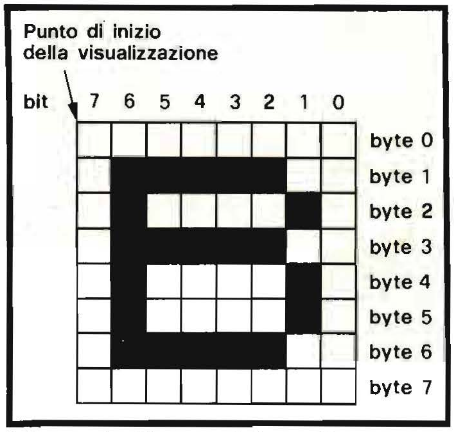


Figura 1. *Composizione di un carattere e punto di inizio per la visualizzazione.*

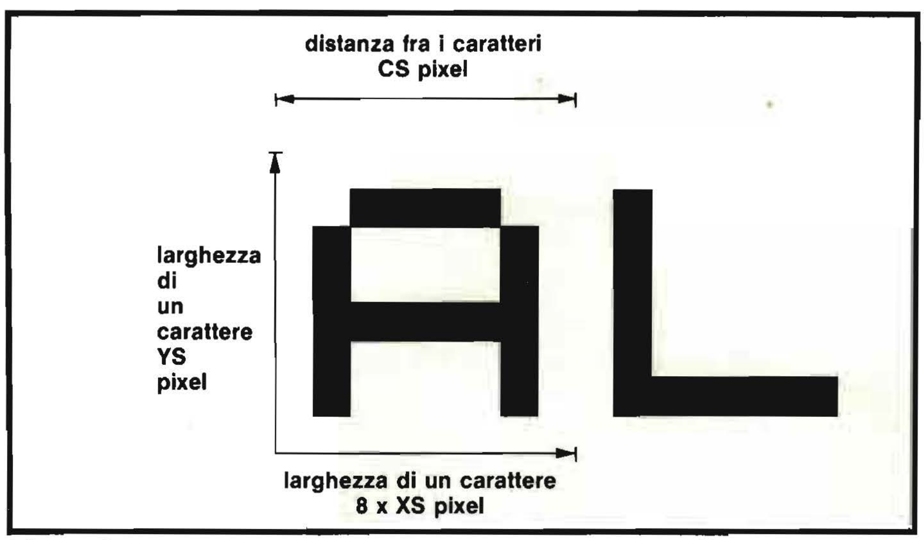


Figura 2. *I vari parametri di ingrandimento adoperati dalla routine.*

Tutti i possessori di uno ZX Spectrum conoscono senza altro la cassetta "Horizon"; si tratta, per quanti non lo sapessero, di una cassetta di software che fa parte della documentazione didattica unita ad ogni confezione dello Spectrum. Il suo contenuto comprende un programma interattivo di descrizione dell'hardware, un corso, articolato in più programmi, sull'uso della tastiera nonché vari programmi dimostrativi che toccano gli argomenti più vari. In tutti questi programmi vengono usati caratteri più o meno ingranditi, per produrre messaggi di notevole effetto. Per la generazione di questi caratteri è utilizzata una routine in linguaggio macchina che viene caricata in coda a ciascun programma. Le sue caratteristiche sono troppo interessanti perché il suo uso resti confinato ai soli programmi della cassetta.

In questo articolo sono indicate le operazioni da effettuare per poter disporre di questa ruotine in ogni situazione, indipendentemente dal tipo di programma con cui verrà usata o della posizione di memoria che occuperà. Infine ne viene presentata un'applicazione particolare, rivolta soprattutto all'utenza "seria" dello Spectrum: l'aumento della capacità del video, fino ad un massimo di 63 colonne per 32 righe, pari a 2016 caratteri, contro i 704 della configurazione normale (32 x 22).

Per prima cosa vediamo in dettaglio quali sono le operazioni compiute dalla routine quando viene chiamata.
Perché questa possa funzionare correttamente è necessario innanzitutto che in un'apposita area di memoria, di cui parleremo più avanti e che per ora ci limiteremo a chiamare area dati, vi siano le seguenti istruzioni:

- la stringa di caratteri da visualizzare;
- le coordinate del punto di inizio della visualizzazione, ossia il primo punto in alto a sinistra della griglia che rappresenta il primo carattere (la figura 1 chiarisce questo concetto);
- il fattore di ingrandimento dei caratteri nel senso della larghezza;
- il fattore di ingrandimento dei caratteri nel senso dell'altezza;
- il fattore di spostamento fra un carattere e il successivo.

La routine si servirà di questi dati per riprodurre i caratteri a partire dalle coordinate date, ingranditi in senso orizzontale e verticale secondo i rispettivi fattori, e distanziati fra loro del fattore di spostamento moltiplicato per il fattore di ingrandimento nel senso della larghezza.
Chiamando:
XS il fattore di ingrandimento in larghezza;
YS il fattore di ingrandimento in altezza;
CS il fattore di spostamento.

Ogni carattere sarà largo 8 x XS

Ingrandimento e riduzione di caratteri

pixel, alto 8 x YS pixel e disterà dal carattere precedente CS x XS pixel; questo perché la griglia che forma un carattere normale, cioè con ingrandimento pari a 1, e di 8 x 8 pixel (figura 2).
Di conseguenza:
- il massimo fattore di ingrandimento possibile per una stringa di N caratteri sarà 256/(N x CS);
- il valore normale per CS è 8; valori inferiori provocano un accorciamento del lato destro di ogni carattere, mentre valori superiori creano degli spazi fra i caratteri.

Tenete inoltre presente che non è prevista la possibilità di andare a capo automaticamente, per cui nel nostro caso si fossero fatti male i conti i caratteri in eccesso andrebbero a coprire quelli all'inizio della stessa riga.

Per quanto riguarda gli attributi, vengono riconosciuti quelli definiti globalmente con PAPER, INK, BRIGHT e FLASH. Quindi caratteri di colore, sfondo e luminosità a vostra scelta, lampeggianti o no, ma sempre in OVER 0 e INVERSE 0 (questo perché le informazioni relative ad OVER e INVERSE non sono contenute né nella variabile di sistema ATTR P, cui la routine fa riferimento, né nei singoli byte degli attributi, su cui la routine opera).

Esaminiamo ora le aree di memoria interessate dalla routine, e le modifiche necessarie per cambiarle.
- Area programma: è quella in cui risiede la routine stessa.

Sarà sempre lunga 277 byte, mentre il suo inizio potrà essere variato a seconda delle esigenze (si usa dire in questi casi che la routine viene rilocata). Occorre tener conto, però, di alcuni indirizzamenti diretti che fanno riferimento a byte del programma e quindi dovranno essere modificati di conseguenza. La tabella 1 elenca le locazioni da modificare, con la posizione del byte cui si riferiscono, indicata usando come riferimento la posizione del primo byte del programma, indicata con X. Se per esempio, vogliamo collocare la nostra routine a partire dalla locazione di memoria 31860, le locazioni X + 86 e X + 87 dovranno contenere il numero 31863, cioè X + 3 come troviamo indicato in tabella; a questo punto dobbiamo calcolare i valori da porre in ciascuno dei due byte, quello meno significativo e quello più significativo: il primo sarà 31863 − 256 ★ INT (31863/256) e il secondo INT (31863/256), da porre rispettivamente nel byte 31946 (X + 86) e 31947 (X + 87).
Riassumiamo con una formula generale, detti:
X + H la locazione del byte più significativo;
X + L la locazione del byte meno significativo;
X + V il valore da assegnare ai due byte.
potremo scrivere:
POKE X + H, INT (X + V)/256)

POKE X + H, INT (X + V)/256)
POKE X + L, X + V − 256 ★ INT (X + V/256)
esprimendoci nel BASIC dello Spectrum.

Caricando la routine dalla cassetta "Horizon", questa potrà avere tre diversi indirizzi di inizio, a seconda del programma cui è associata: 32000, 32196 e 32256; per sapere a quale di questi è stata caricata; basta andare a vedere quale è l'argomento dell'istruzione CLEAR usata per riservarle lo spazio: l'indirizzo di inizio sarà il byte successivo. Questo vale anche se il programma utilizza altre routine in linguaggio macchina, poiché quella che ci interessa viene sempre collocata per prima (per trovare l'istruzione CLEAR tenete presente che questi programmi sono stati salvati in modo da partire con una delle ultime linee, quindi intorno alla 9000-9500: è in quei paraggi che dovrete cercarla). Una volta accertato l'indirizzo di inizio, a voi la scelta: lasciare la routine dove si trova, senza quindi doverla modificare, o rilocarla applicando le modifiche di cui abbiamo parlato.
- Area dati: è quella che contiene tutte le informazioni provenienti dall'esterno ed indispensabili al corretto funzionamento della routine; viene elaborata ed aggiornata durante l'uso.

In origine è stata destinata a questo scopo l'area "Printer Buffer" ossia quella usata dalla stampante ZX, che si estende dall'indirizzo 23296 all'indirizzo 23552. In realtà, non essendo possibile operare su più di una singola riga, vengono utilizzati solo 39 byte, e precisamente quelli dal 23296 al 23335.
Anche un uso così limitato rende comunque inutilizzabile questa area in modo contemporaneo dalla routine e dalla stampante. Se il vostro programma fa uso della ZX Printer, potete aggirare l'ostacolo in due modi: "ripulendo" l'area dopo ciascun uso, con una sequenza del tipo:

```
FOR i = 23296 TO 23552
POKE i,0
NEXT i
```

od utilizzando un'altra zona di memoria come area dati.

Sebbene quest'ultima soluzione sia sconsigliabile, per le troppe locazioni da modificare, la tabella 2 dà le istruzioni necessarie a questa operazione, riferendo i byte del programma alla posizione X del primo, e i byte dell'area dati alla posizione Y del primo (per intenderci, quello che originariamente è in posizione 23296). Usando la stessa terminolo-

gia di prima, avremo
POKE X + H, INT ((Y + V)/256)
POKE X + L, Y + V − 256 ★ INT ((Y + V)/256)

In figura 3 troviamo invece l'organizzazione dell'area stessa, che contiene tutte le variabili di cui abbiamo parlato all'inizio più un'area di lavoro usata per memorizzare prodotti intermedi dell'elaborazione. Per introdurre i dati necessari nel giusto ordine possiamo senz'altro ricorrere ad una serie di POKE, ma questa serie di istruzioni è certamente più comoda per trasferire nell'area variabili BASIC:

LET a = 23306
POKE a, x: POKE a + 1, y: POKE a + 2, xs: POKE a + 3, ys: POKE a + 4, cs
LET a = a + 4: LET w = LEN p$
FOR m = 1 TO w: POKE a + m, CODE pS (m): NEXT m
POKE a + w + 1,255

Nel caso abbiate modificato l'area dati, al posto di 23306 metterete l'indirizzo di inizio dell'area che avete scelto.

• Variabili di sistema.

Vengono usate le variabili CHARS e ATTR P, la prima per poter attingere al set dei caratteri, la seconda per conoscere quali attributi sono stati definiti. Può essere interessante utilizzare un set di caratteri alternativo, magari formato dalla grafica definibile, ed in questo caso è possibile l'uso contemporaneo dei due set, uno nelle PRINT normali, l'altro utilizzando la routine. Per far questo non bisogna modificare il valore di CHARS, come si farebbe normalmente per cambiare set: è sufficiente modificare, all'interno della routine, quello che viene preso come l'indirizzo di CHARS.
In questo modo avremo due CHARS, una per il BASIC e l'altra solo per la routine. I byte che contengono la posizione di CHARS hanno indirizzi X + 18 e X + 19 (rispettivamente, byte meno significativo e byte più significativo); in essi dovremo introdurre l'indirizzo

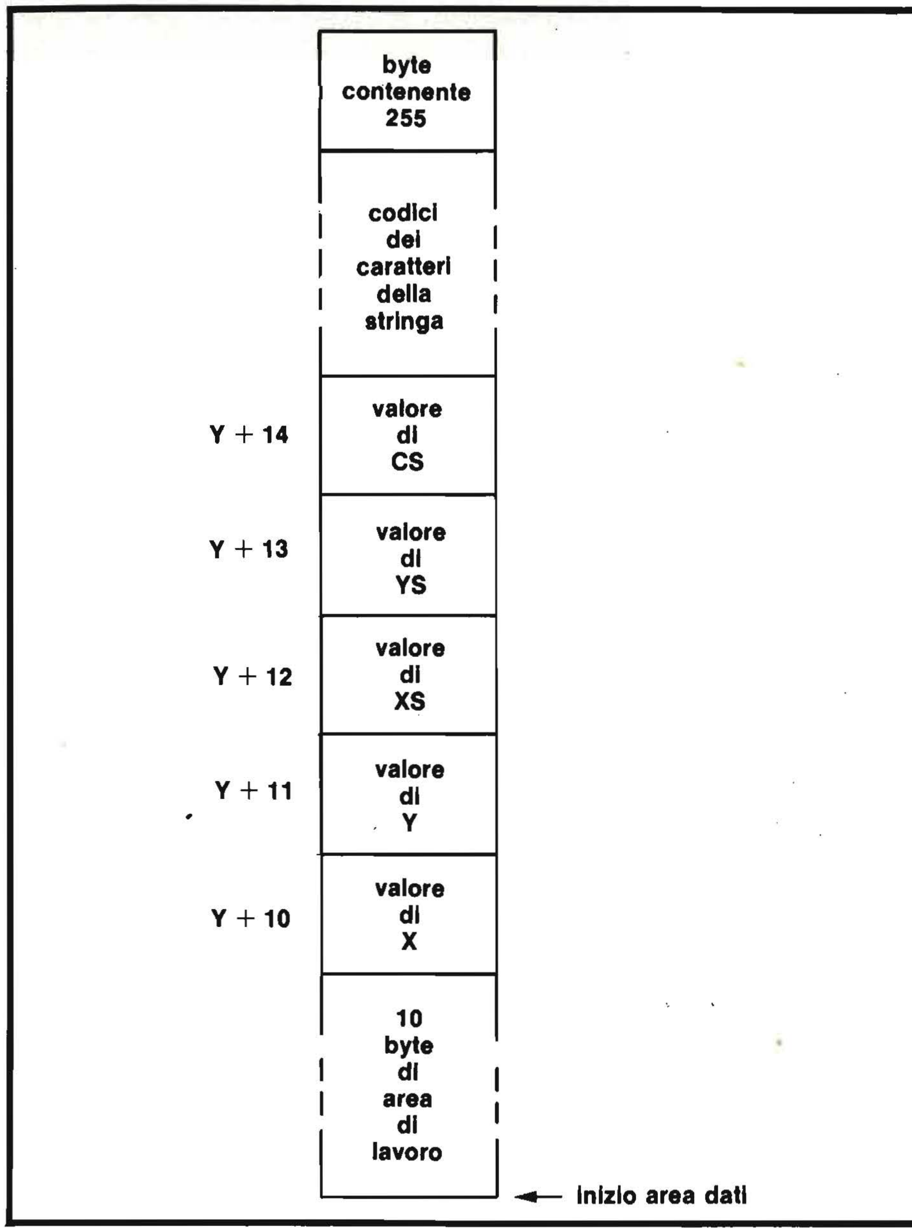


Figura 3. *Organizzazione dell'area dati della routine.*

del primo di due byte in cui avremo posto l'indirizzo d'inizio del set diminuito di 256 (per maggiori chiarimenti leggete quanto viene detto a proposito di CHARS nel manuale, al capitolo dedicato alle variabili di sistema).

Infine due parole sul metodo da usare per venire praticamente in possesso della routine. I passi da compiere sono:
1) caricare uno qualsiasi dei programmi contenuti nella cassetta "Horizon";
2) conoscere l'indirizzo di inizio della routine;
a questo punto, se non avete bisogno di rilocarla, basta dare NEW per eliminare il programma e il gioco è fatto. Volendo invece cambiare la posizione, occorre operare tutte le modifiche viste sopra, e quindi spostarla con istruzioni del tipo:
LET a = vecchio indirizzo di inizio
LET b = nuovo indirizzo di inizio
FOR i = 1 TO 277
POKE b − 1 + i, PEEK (a − 1 + i)
NEXT i
e quindi
3) dare CLEAR b − 1;
4) dare NEW.

Per utilizzarla basta ora inserire nel programma in posizione opportuna, che normalmente sarà subito dopo il gruppo di istruzioni a caricare l'area dati, una istruzione:

RANDOMIZE USR
indirizzo di inizio

o, meglio:
LET w = USR indirizzo di inizio
dove w è una variabile che non ci serve; il vantaggio di questa forma

Ingrandimento e riduzione di caratteri

```
   5 LET start=inizio routine:
     LET char=inizio nuovo set
  10 FOR i=char TO char+28*8-1:
     READ a: POKE i,a: NEXT i
  20 POKE start+18,23728-256*INT
     (23728/256)
  30 POKE start+19,INT (23728/
     256)
  40 POKE 23728,char-65*8-256*
     INT ((char-65*8)/256)
  50 POKE 23729,INT ((char-65*8
     )/256)-1
  60 LET xs=1: LET ys=1: LET cs
     =4
  70 LET x=0: LET y=0
  80 PAUSE 0
  90 LET p$=INKEY$
 100 GO SUB 900
 110 LET x=x+4
 120 IF x=252 THEN LET x=0: LET
     y=y+8
 130 GO TO 80
 900 LET a=23306
 910 POKE a,x: POKE a+1,y: POKE
     a+2,xs: POKE a+3,ys: POKE
     a+4,cs
 920 LET a=a+4: LET w=LEN p$
 930 FOR m=1 TO w: POKE a+m,
     CODE p$(m): NEXT m
 940 POKE a+w+1,255: LET w=USR
     start
 950 RETURN

1000 DATA 0,64,160,160,224,160,160,0
1010 DATA 0,192,160,192,160,160,224,0
1020 DATA 0,64,160,128,128,160,64,0
1030 DATA 0,192,160,160,160,160,192,0
1040 DATA 0,224,128,192,128,128,224,0
1050 DATA 0,224,128,192,128,128,128,0
1060 DATA 0,64,160,128,128,160,95,0
1070 DATA 0,160,160,224,160,160,160,0
1080 DATA 0,224,64,64,64,64,224,0
1090 DATA 0,32,32,32,32,160,64,0
1100 DATA 0,160,160,192,160,160,160,0
1110 DATA 0,128,128,128,128,128,224,0
1120 DATA 0,160,224,160,160,160,160,0
1130 DATA 0,224,160,160,160,160,160,0
1140 DATA 0,224,160,160,160,160,224,0
1150 DATA 0,224,160,160,224,128,128,0
1160 DATA 0,224,160,160,160,160,224,64
1170 DATA 0,224,160,160,192,160,160,0
1180 DATA 0,96,128,64,32,160,64,0
1190 DATA 0,224,64,64,64,64,64,0
1200 DATA 0,160,160,160,160,160,224,0
1210 DATA 0,160,160,160,160,160,64,0
1220 DATA 0,160,160,160,160,224,160,0
1230 DATA 0,160,160,64,160,160,160,0
1240 DATA 0,160,160,64,64,64,64,0
1250 DATA 0,224,32,64,128,128,224,0
```

Listato 1. *Programma per la creazione di un video 63 x 24.*

XS	YS	CS
1	2	8
2	2	7
2	2	8
2	3	8
3	3	8
4	4	8
6	6	8

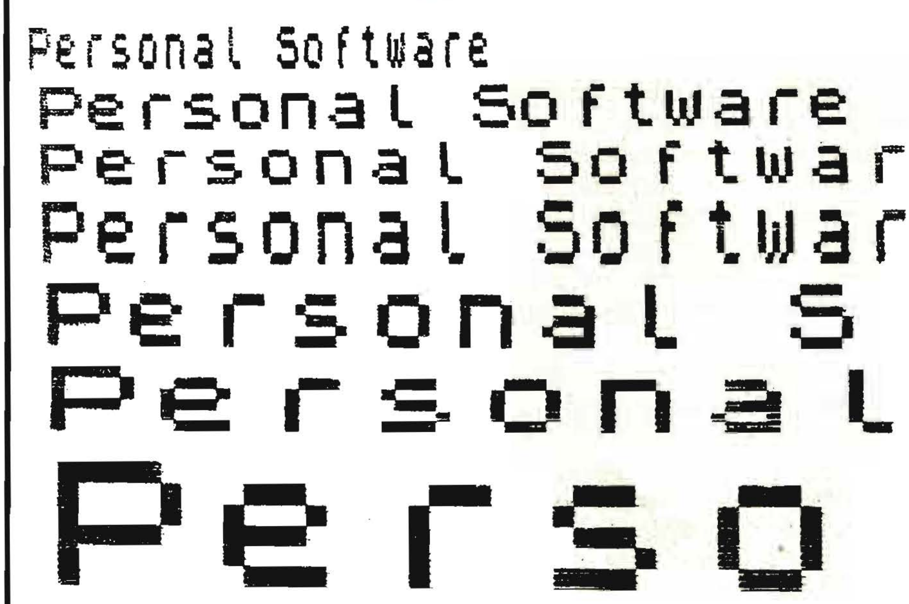


Figura 4. *Esempi di vari ingrandimenti operati sui medesimi caratteri.*

```
 MA PRIMA FARO' ALCUNA ESPERIENZA, AVANTI CH'IO PIU' OLTRE PRO
CEDA, PERCHE' MIA INTENZIONE E' ALLEGARE PRIMA L'ESPERIENZA, E
POI COLLA RAGIONE DIMOSTRARE, PERCHE' TALE ESPERIENZA E'
COSTRETTA IN TAL MODO AB OPERARE.
 E QUESTA E' LA VERA REGOLA, COME LI SPECULATORI DELLI EFFETTI
NATURALI HANNO A PROCEDERE, E ANCORA CHE LA NATURA COMINCI DAL
LA RAGIONE E TERMINI NELLA SPERIENZA, A NOI BISOGNA SEGUITARE
IN CONTRARIO,CIOE' COMINCIANDO, COME DI SOPRA DISSI  DALLA SPE
RIENZA, E CON QUELLA INVESTIGARE LA RAGIONE

                LEONARDO DA VINCI  DAL CODICE ATLANTICO
```

Figura 5. *Hard copy di uno schermo 63 x 24 con un brano di testo.*

rispetto alla precedente è che quest'ultima non condiziona la generazione di numeri casuali, consentendo così l'uso di RND.

Eccoci ora ad un uso molto particolare di questa routine: l'aumento della capacità video.
I programmi che richiedono all'utilizzatore un confronto visivo dei dati, siano essi numeri (in programmi tipo VU-...) o testo (in programmi di Word Processing), perdono gran parte della loro efficacia d'uso quando sono costretti in un piccolo schermo.

Quello che ci proponiamo di ottenere è una modifica dell'immagine (l'eco) dei caratteri sullo schermo, per renderli più piccoli, senza modificare i loro codici, in modo da mantenere inalterata la possibilità di utilizzare stampanti non ZX od altre periferiche cui vengono appunto inviati i codici dei caratteri, che pertanto devono essere rigorosamente rispondenti allo standard ASCII. Senza questa condizione verrebbe meno lo scopo principale di programmi del tipo di quelli citati, che è appunto una stampa di una certa qualità.
L'aumento della densità dei caratteri sul video si articola in vari passaggi:
- creazione in memoria di un set in cui ciascun carattere occupi soltanto i primi n bit in laghezza e m bit verso il basso della griglia 8 x 8 disponibile;

Ingrandimento e riduzione di caratteri

• uso della routine di cui sopra con un incremento di X pari ad n, e comunque inferiore ad 8, ed un incremento di Y ad ogni riga pari ad m.

Come vedete in questo caso non utilizziamo le capacità di ingrandimento della routine, ma solo la possibilità di visualizzare i caratteri senza essere legati alle righe e colonne previste. Utilizzando per le dimensioni dei caratteri dei sottomultipli di 256 e 192 si evitano sprechi di spazio. L'ultimo carattere di ciascuna riga non va invece utilizzato, perché la sua "appendice" andrebbe a cancellare il primo carattere della riga stessa (in tutti i caratteri non sono al termine di riga questa appendice viene coperta dal carattere successivo, per effetto del ridotto incremento di X).

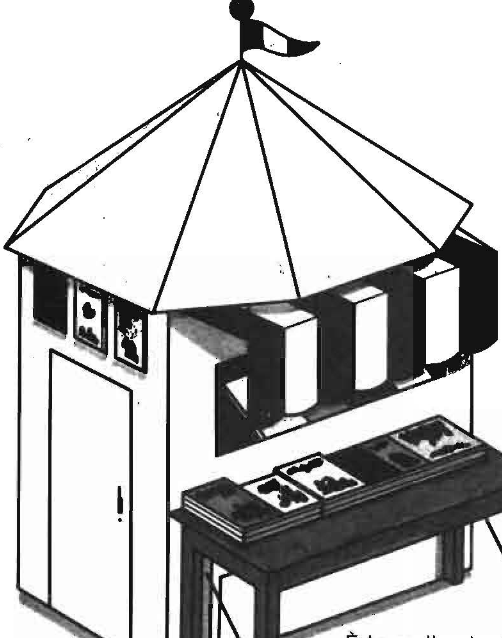
Il programma presentato nel listato 1 dimostra la possibilità di ottenere uno schermo 63 x 24; collocando in linea 5 l'indirizzo di inizio della routine e del nuovo set, è pronto a funzionare. Come variabile CHARS fittizia utilizza gli indirizzi 23728 e 23729, che pur essendo fra le variabili di sistema non vengono utilizzati, e per questo modifica di conseguenza gli indirizzi start + 18 e start + 19 della routine. In questa CHRS viene introdotto l'indirizzo di inizio dell'area dati, cioè la variabile char, diminuito di 256. Le linee 1000-1250 contengono i dati per costruire in memoria, all'indirizzo da voi specificato, il nuovo set; trattandosi di un programma esemplificativo questo è limitato alle lettere, dalla A alla Z. Notate come l'indirizzo contenuto nella nostra variabile CHARS sia spostato indietro, rispetto al reale inizio del set, di 65 caratteri: infatti la A, primo carattere da noi collocato in memoria, è il 97 che meno 32 (numero di codici del primo carattere compreso nel set normale) dà appunto 65. Certo l'aspetto grafico dei nuovi caratteri non è meraviglioso, ma occorre tener presente che tutto quello che ci serve è che siano leggibili: l'aspetto elegante lo avremo eventualmente nella stampa, e dipenderà dal tipo di stampante utilizzata. Il medesimo programma, con poche modifiche, può essere usato per produrre un video 63 x 32, dimezzando l'altezza dei caratteri; a questo punto, comunque, siamo veramente al limite della visibilità, tenendo anche conto del fatto che non tutti i televisori danno un'immagine ugualmente nitida (quanti sono i fortunati che posseggono un monitor?). ■

Indirizzi di programma da modificare		Valore da introdurre
byte meno significativo	byte più significativo	
X + 86	X + 87	X + 3
X + 106	X + 107	X + 32
X + 127	X + 128	X + 164
X + 154	X + 155	X + 48
X + 251	X + 252	X + 156
Gli indirizzi sono riferiti all'indirizzo di inizio del programma.		

Tabella 1. *Elenco delle locazioni da modificare per rilocare la routine.*

Indirizzi di programma da modificare		Valore da introdurre
byte meno significativo	byte più significativo	
X + 1	X + 2	Y + 15
X + 6	X + 7	Y + 0
X + 24	X + 25	Y + 4
X + 27	X + 28	Y + 11
X + 30	X + 31	Y + 9
X + 33	X + 34	Y + 10
X + 36	X + 37	Y + 8
X + 41	X + 42	Y + 5
X + 46	X + 47	Y + 2
X + 50	X + 51	Y + 6
X + 53	X + 54	Y + 5
X + 59	X + 60	Y + 4
X + 65	X + 66	Y + 14
X + 69	X + 70	Y + 12
X + 73	X + 74	Y + 10
X + 80	X + 81	Y + 10
X + 83	X + 84	Y + 0
X + 89	X + 90	Y + 4
X + 92	X + 93	Y + 13
X + 96	X + 97	Y + 9
X + 100	X + 101	Y + 9
X + 103	X + 104	Y + 2
X + 109	X + 110	Y + 5
X + 112	X + 113	Y + 12
X + 116	X + 117	Y + 9
X + 119	X + 120	Y + 7
X + 122	X + 123	Y + 13
X + 131	X + 132	Y + 7
X + 136	X + 137	Y + 7
X + 142	X + 143	Y + 8
X + 146	X + 147	Y + 8
X + 152	X + 153	Y + 6
X + 177	X + 178	Y + 8
X + 183	X + 184	Y + 7
X + 217	X + 218	Y + 7
X + 240	X + 241	Y + 8
X + 264	X + 265	Y + 6
Gli indirizzi sono riferiti all'indirizzo di inizio del programma e dell'area dati.		

Tabella 2. *Elenco delle locazioni da modificare per utilizzare una diversa area dati.*



è in edicola

L. 8.000

È in edicola la prima **Guida all'acquisto di libri italiani ed esteri principalmente di informatica, di elettrotecnica ed elettronica, nonché del software applicativo.**
Oltre 350 testi italiani, 1000 stranieri in lingua originale e 900 package applicativi costituiscono l'attuale assortimento.
La guida è il tuo consulente sicuro per orientarsi nel labirinto dell'editoria tecnica, lo strumento ed il servizio essenziale per chi ha compreso l'importanza della tecnologia nel mondo odierno.
Libri di base e didattici per imparare e capire; applicativi per realizzare e coltivare il proprio hobby; pratici per risolvere i problemi dell'attività quotidiana; di elevata specializzazione per migliorare il proprio background professionale o culturale.
Software per Apple, IBM, Texas, Sinclair, TRS, VIC per risolvere i problemi più complessi o, semplicemente, per giocare. Un'ampia gamma di "applicativi" che comprende tra gli altri i più efficienti Data Base, i più completi programmi per l'elaborazione dei testi, i più sofisticati package grafici oltre naturalmente, ai più divertenti programmi ricreativi.
E inoltre, la nuova linea Software TechnoClub, sviluppata in collaborazione con programmatori professionisti, con una gamma di programmi selezionati e convenienti per il tuo home o personal computer.
Acquista la guida in edicola o ordinala direttamente, compilando e spedendo il coupon sottoriportato, unitamente a L. 8.000.
Potrai così prendere visione anche delle modalità per diventare Socio del **TechnoClub** e godere dei numerosi vantaggi che ne derivano tra i quali
La ricezione gratuita di minimo 8 ulteriori numeri di questa guida. Sarai così costantemente aggiornato su tutte le novità editoriali più qualificate e sui package più interessanti ed innovativi per il tuo computer.
Nessun impegno di acquisto durante il periodo di adesione. Scelta libera e senza vincoli di minimi quantitativi di acquisto durante il periodo di adesione, potendo così ordinare ciò che si vuole, quando si vuole.
Convenienza certa. I testi italiani sono scontati del 10% circa rispetto al prezzo di copertina.
Particolarmente vantaggiosi risultano i prezzi dei libri esteri e del software.
La tessera TechnoClub. Il documento personale che dà diritto a sconti speciali su diversi articoli acquistati presso negozi convenzionati.
Oltre 5.000 Soci hanno già aderito al TechnoClub. Attendiamo anche te.



Ordino il primo numero della
Guida all'acquisto di libri e software e

☐ allego L. 8.000 (in contanti o francobolli)

☐ allego assegno di L. 8.000

Nome ______________________

Cognome ______________________

Via ______________________

Città ______________ C.A.P. __________ Prov. ______

Data ______________________

Firma ______________________

CEDOLA da compilare e spedire in busta chiusa a Gruppo Editoriale Jackson srl Via Rosellini, 12 20124 Milano

SHARP

La mappa della memoria del PC-1251

di Mauro Lenzi

Nel numero precedente di questa rubrica abbiamo scoperto tre istruzioni BASIC presenti nel computer Sharp PC-1251 e non dichiarate sul manuale: PEEK, POKE e CALL.

Con queste istruzioni siamo già in grado di sondare la memoria e trovare cose molto interessanti e, come vedremo, piuttosto "strane".

Facciamo girare il seguente programma:

```
10  L = 0
20  POKE L,92
30  IF PEEK L = 92 THEN BEEP 3 : STOP
40  L = L + 1: GOTO 20
```

Finalmente, dopo una lunga attesa, sentiamo i tre beep ed il programma si arresta. Il valore di L in cui il computer si è arrestato è 32768, corrispondente a 2^{15}: abbiamo così trovato il numero corrispondente alla prima locazione della RAM.

Il metodo utilizzato è semplicissimo: si basa sull'ovvio principio che le locazioni della memoria ROM non possono essere cambiate nemmeno con l'istruzione POKE ed il loro contenuto rimane quindi invariato anche dopo avere eseguito questa istruzione.

A questo punto sarebbe logico supporre che tutta la memoria da 0 a 32767 faccia parte della ROM, tuttavia, poiché il manuale dichiara che vi sono solo 24 Kbyte di ROM, occorre indagare più a fondo.

Eseguiamo dunque il seguente programma:

```
10  L = 0
20  PAUSE PEEK L
30  L = L + 256 : GOTO 20
```

Il display si limita a mostrarci una lunga successione di numeri da 0 a 63 e poi visualizza dei numeri "casuali". Con qualche altra breve istruzione è facile trovare, con esattezza, che tutta la memoria compresa fra 0 e 16383, pari a 2^{14}, contiene una serie di numeri che vengono incrementati di uno ogni 256 locazioni.

Proseguendo il sondaggio, si scopre facilmente che tutti gli indirizzi da 32768 a 50900 possono essere agevolmente cambiati. Ci troveremmo dunque a disposizione più di 18 Kbyte di RAM. Sono spiacente di deludere le vostre speranze, ma non è così. Infatti, facciamo un'analisi più approfondita, troviamo il trucco; esaminiamo le seguenti locazioni:

```
PEEK 32768
PEEK 36864
PEEK 40960
PEEK 45056
```

Compare sempre lo stesso numero. Digitiamo ora:

```
POKE 45056,92
```

Se andiamo ad analizzare nuovamente le locazioni precedenti, troveremo che anche in esse vi è il numero 92! La spiegazione è semplice: le aree di memoria da 32768 a 36863, da 36864 a 40959, da 40960 a 45055 e da 45056 a 49151 sono coincidenti. La nostra memoria RAM dunque si riduce a 5,8 Kbyte, cioè da 45056 a 50900, che sono sempre un po' di più dei 4 Kbyte dichiarati.

In particolare si può trovare che le locazioni da 45056 a 45103 contengono la memoria del RSV mode, mentre quelle da 45104 a 48590 sono riservate a quello che in "gergo" viene chiamato Text Buffer, che altro non è se non il nostro programma codificato.

La Symbol Table, cioè le locazioni in cui vengono memorizzati i valori delle variabili, parte da 50846 e da lì scende progressivamente: il valore di A o di A$ va da 50840 a 50846, il valore di Z viene memorizzato da 50640 a 50646; A(255) inizia alla locazione 48808.

In figura 1 è stata raffigurata la màppa della memoria del nostro Sharp PC-1251.

Della parte di memoria che va da 50847 a 65535, almeno per il momento non ce ne occuperemo, perché mescolando locazioni RAM a locazione ROM è di difficilissima comprensione, senza avere a disposizione il manuale del firmware del computer. Comunque con le novità che abbiamo appreso, la prossima volta faremo cose interessanti come trovare ed usare con disinvoltura caratteri "strani" ad esempio lettere greche, "cinesi e misteriosi simboli matematici. Dulcis in fundo, scopriremo altre 6 istruzioni sconosciute. ■

LOCAZIONI da	LOCAZIONI a	DESCRIZIONE
0	16383	*Numeri progressivi da 0 a 63.*
16384	32767	*ROM.*
32768	36863	*Quattro blocchi di memoria RAM coincidenti (numero di default = 4096).*
36864	40959	
40960	45055	
45056	49151	
45056	50900	*RAM utilizzabile.*
45056	45103	*Memoria del Reserve Mode.*
45104	50846	*Text Buffer + Symbol Table.*
50847	65535	*Utilizzate dal Sistema Operativo, di difficile interpretazione.*

Figura 1. *Schema esemplificativo della distribuzione della memoria nel computer Sharp PC-1251.*

ZX SPECTRUM

Come incolonnare correttamente i numeri senza ricorrere a una subroutine

di Marcello Spero

Tutti i linguaggi ad alto livello prevedono la possibilità di definire il formato di stampa, cioè il modo in cui verranno disposti, sullo schermo o sulla carta, i dati in uscita.

Per quanto riguarda i dati numerici, in particolare, esistono parametri che ne definiscono la lunghezza massima, il numero di decimali e se sono di tipo reale. In base a queste specifiche la macchina provvederà ad un corretto incolonnamento secondo le regole dell'algebra e cioè con gli interi allineati a destra e i reali allineati sul punto decimale.

STAMPA LIBERA	STAMPA A FORMATO
5	5
100	100
3375	3375
160000	160000
5	5.
100	100.
3375	3375.
160000	160000.
4	4.00
64	64.00
1728	1728.00
65536	65536.00
1.2	1.200
4.16	4.160
27.216	27.216
256.4096	256.410
0.999487	0.9995
0.999687	0.9997
.014285714	.01429
.00081632653	.00082
.000078717201	.00008
.000010662224	.00001

Figura 1. *Alcuni esempi di stampa normale e facendo uso delle funzioni.*

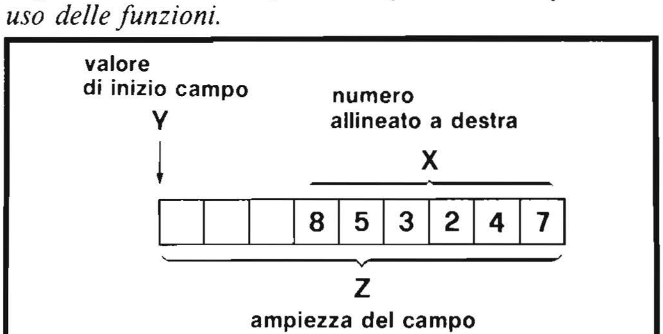


Figura 2. *Struttura di un campo intero.*

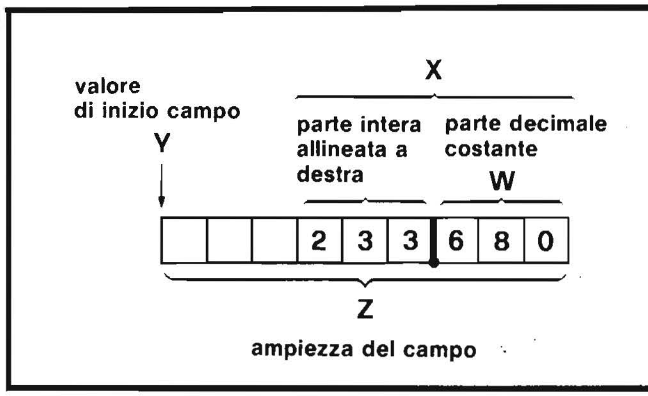


Figura 3. *Struttura di un campo decimale.*

I vantaggi di questo procedimento non si limitano ad una migliore leggibilità di lunghe colonne di numeri. Nel caso di dati reali, infatti, averne fissato il numero di decimali comporta un'aggiunta di zeri qualora questi siano meno del previsto e l'arrotondamento dell'ultima cifra nel caso opposto, rendendo così evidente il grado di precisione che abbiamo voluto attribuire ai dati in questione.

Per quanto riguarda lo ZX Spectrum, troviamo nel suo BASIC le istruzioni per il posizionamento della stampa (TAB, AT) ma non quelle per la definizione del formato dei dati numerici (normalmente attuato in BASIC con una istruzione PRINT USING).

Per creare un'azione di questo tipo si è fatto uso della versatile istruzione DEF FN, definendo in tal modo alcune funzioni in grado di controllare l'output numerico. Rispetto all'uso di una subroutine vera e propria questo sistema ha due vantaggi: la semplicità d'uso e il non essere legato a determinate variabili per il passaggio dei parametri; d'altra parte uno svantaggio può essere l'uso di condizioni implicite, le uniche utilizzabili all'interno delle DEF FN, che hanno reso più macchinoso il procedimento.

Esaminiamo il primo caso, e cioè la creazione di un "campo intero" ossia uno spazio, ampio un certo numero di caratteri, all'interno del quale verrà collocato un dato intero, allineato a destra. La funzione interessata è in questo caso FN i (x,y,z,), dove x sarà il dato, y il valore di TAB di inizio campo e z l'ampiezza del campo stesso. All'istruzione;

```
PRINT TAB y;x
```

usata normalmente andrà sostituita una:

```
PRINT TAB FN i(x,y,z);x
```

in cui x, y e z potranno essere costanti o variabili qualsiasi.

A seconda dell'ampiezza di campo prescelta potranno presentarsi tre casi:
1) le dimensioni del campo sono superiori a quelle del dato, che verrà quindi allineato a destra.
2) il dato ha la stessa lunghezza del campo
3) le dimensioni del campo sono inferiori a quelle del dato; in questo caso la condizione implicita posta alla fine della funzione addiziona al valore di TAB un numero (−100) tale da renderlo sicuramente negativo

Come incolonnare correttamente i numeri senza ricorrere a una subroutine

e provocare così la condizione di errore "B Integer out of range". Questo meccanismo di sicurezza è utile per segnalare la presenza di dati anomali, ma può essere facilmente eliminato togliendo dalla funzione tutta l'ultima parentesi, appunto il controllo di lunghezza. In questo modo dati più lunghi del campo prescelto verranno rappresentati ugualmente, allineati con gli altri ma sporgenti sulla sinistra oltre il limite che avevamo fissato.

Passiamo ora alla formattazione di dati reali. Il procedimento è più complesso, trattandosi non solo di allineare il dato all'interno di un campo, ma di rendere costante il numero dei suoi decimali mediante troncamento e arrotondamento o aggiunta di zeri. Queste operazioni vengono svolte da tre funzioni diverse: FN r(x,y,z,w), FN t$(x,w) e FN a$(x,w) la prima delle quali provvede all'allineamento mentre le altre due all'arrotondamento o aggiunta di zeri. Per poterle utilizzare correttamente, all'istruzione:

PRINT TAB y;x

andrà sostituita una:

PRINT TAB FN r(x,y,z,w); FN t$(x,w)

dove x è il dato, y il valore di TAB di inizio campo, z l'ampiezza del campo e w il numero di decimali che si desidera compaiano nella stampa.

Osserviamo ora più da vicino il funzionamento di ciascuna funzione. FN r si occupa, come abbiamo detto, dell'allineamento del dato all'interno del campo reale. Questo tipo di campo si può considerare diviso in due settori, uno per la parte intera e l'altro per i decimali. All'interno del primo settore l'allineamento avviene come per il campo intero, ma occorre tenere conto di una limitazione insita nel BASIC dello Spectrum. I numeri inferiori a 0.1, infatti, sono rappresentati senza lo zero iniziale, e quindi la loro parte intera non esiste; è impossibile, oltretutto, l'aggiunta di un carattere "0" alla FN t$, pur essendo questa una funzione stringa, per motivi non chiari ma legati alla meccanica della DEF FN. FN r, comunque, tiene conto di questo ed incolonna correttamente sia i numeri senz'altro minori di 0.1 che quelli, come 0.999999... che pur essendone in origine minori dopo l'arrotondamento operato, come vedremo in seguito, da FN t$, ne risulteranno maggiori, per mezzo del fattore di compensazione rappresentato da 10↑(-w-1). Anche qui abbiamo una condizione di errore nel caso la parte intera del dato superi l'ampiezza del primo settore del campo, ed anche qui è possibile la sua eliminazione togliendo dalla FN r l'ultima parentesi.

Il secondo settore, invece, deve essere riempito

```
1 DEF FN i(x,y,z)=y+z-LEN
  STR$ x-(100 AND LEN STR$ x>
  z)

2 DEF FN r(x,y,z,w)=y+z-w-
  (ABS x>=.1-10↑(-w-1))-LEN
  STR$ INT x+(-100 AND LEN
  STR$ INT x+w+1>z)

3 DEF FN a$(x,w)=STR$ (x+5*
  10↑(-w-1)*(-1 OR x>=0))

4 DEF FN t$(x,w)=FN a$(x,w)
  ( TO LEN STR$ INT x+w+(ABS
  x>=.1-10↑(-w-1)))


5 REM ***********************
      condensazione di FN a$
      e FN t$ in un'unica
      funzione
      ***********************

6 DEF FN t$(x,w)=(STR$ (x+5*
  10↑(-w-1)*(-1 OR x>=0)))
  ( TO LEN STR$ INT x+w+(ABS
  x>=.1-10↑(-w-1)))
```

Listato 1. *Le funzioni di formattazione.*

completamente, volendo mantenere fisso il numero dei decimali. Di questo si occupa FN t$ che, per la fase di arrotondamento, si avvale di FN a$.

L'arrotondamento consiste nell'addizionare 5 ★ 10 ↑ (− w − 1), che equivale ad aggiungere 5 al primo decimale eccedente, se esiste, o, in caso contrario, ad aggiungere degli zeri al posto dei decimali mancanti. Del troncamento dei decimali eccedenti, che dopo la somma operata da FN a$ saranno sempre almeno uno, si occupa FN t$ operando uno "slicing", cioè una partizione, sul prodotto di FN a$. Le operazioni svolte da FN a$ e da FN t$ possono, volendo, essere riunite in una sola funzione, come vedete in figura 1, ottenendo un insieme peraltro piuttosto ingombrante ma funzionante correttamente.

Per concludere, due parole sull'uso pratico. Una volta inserite nel programma le istruzioni DEF FN basta inserire nelle PRINT, modificate secondo i criteri visti sopra, i parametri corretti.

Ricordo solo che il valore di y deve essere quello della posizione di inizio, cioè dell'estremità sinistra, del campo, e z è sempre, anche nel caso di campi reali, l'ampiezza totale del campo; ne consegue che la differenza fra z e w dovrà essere sufficiente a contenere la parte intera dei dati da stampare, pena il continuo bloccaggio del programma. Molte sono le variazioni possibili: usando FN r e FN t$ con dati interi si ottengono tanti zeri quanti sono i decimali richiesti; ponendo w a zero si avranno sempre numeri interi, ma con il punto decimale; usando la sola FN r si ottiene il solo incolonnamento; con la sola FN t$, infine, si avrà solo il troncamento o arrotondamento dei dati. ■

TI99/4A

Il movimento in TI Extended BASIC

di Sergio Borsani

Ricordo con simpatia le parole dell'amico di Milano: "mi tuffo nell'Assembler e non riemergerò che tra qualche mese". Condivido il disappunto di quanti vorrebbero eseguire in BASIC programmi dove la rapidità dell'azione e la molteplicità degli oggetti in movimento giocano un ruolo determinante ma credo che non si debbano sottovalutare le istruzioni grafiche relative agli sprite fornite dal Modulo TI Extended BASIC. Semmai io sento una limitazione nell'impossibilità di poter usare, dal BASIC, il video in Bit-Map e nella mancanza di un comando CALL LINE per tracciare linee.

Evidentemente queste diverse esigenze sono dovute al fatto che generalmente scrivo programmi didattici e per me la velocità non è essenziale; ma anche chi preferisce dedicarsi ai video game potrà ottenere con l'Extended BASIC risultati soddisfacenti senza dover ricorrere, come con altri computer, alle istruzioni PEEK e POKE per accedere direttamente alle locazioni di memoria o, addirittura, "tuffarsi" nell'Assembler.

Le subroutine alle quali mi riferisco sono SPRITE, MOTION, LOCATE, POSITION, DISTANCE, COINC e DELSPRITE. Le prerogative più evidenti sono il poter determinare la posizione di un oggetto in pixel, senza essere vincolati alle 24 righe e 32 colonne, ed il movimento automatico degli sprite.

Per creare il movimento in Extended BASIC si può procedere essenzialmente in due modi: variare successivamente la posizione con l'istruzione CALL LOCATE o variare la velocità con il comando CALL MOTION. Nel primo caso il moto risulterà poco uniforme, cioè a scatti, soprattutto a velocità elevate; in compenso risulterà molto preciso in quanto si ha continuamente sotto controllo la posizione dello sprite. Nel secondo caso si otterrà un movimento più uniforme ma facilmente si perderà il controllo dello sprite. La ragione di ciò è che, dal momento in cui si determinano le nuove componenti della velocità fino a qundo viene data l'istruzione CALL MOTION, lo sprite continua a spostarsi automaticamente lungo la vecchia direzione; pertanto la variazione di velocità giunge sempre con un lieve ritardo rispetto al punto previsto.

Per un confronto tra i due metodi digitate i brevi listati 1 e 2. In entrambi si vuole ottenere una traiettoria circolare ed apparentemente le differenze sono trascurabili, ma provate, nel secondo, a modificare la linea 240 con V = − 30; non solo è aumentata la velocità ma la traiettoria si è trasformata da circolare a elittica. Per V = −40 lo sprite, addirittura, sfugge al controllo ed esce dal video.

Per ottenere una traiettoria circolare avremmo dovuto cambiare la velocità ad ogni istante; noi purtroppo siamo costretti ad operare ad intervalli discreti. Il listato 3 è un esempio, se vogliamo banale, di come il movimento possa essere controllato con l'uso degli joystick.

Per un uso corretto non deve essere premuto il tasto ALPHA LOCK. Il programma simula il gioco del rincorrersi: un omino deve raggiungere l'avversario che cerca di sottrarsi e quando ciò avviene compare la scritta "PRESO!". L'istruzione CALL COINC, alla linea 210, controlla se c'è la coincidenza tra i due sprite. In caso affermativo la variabile CO assume il valore − 1, in caso contrario la stessa variabile assume il valore 0.

La posizione dello sprite si identifica con quella del pixel in alto a sinistra della piccola superficie quadrata che contiene lo sprite stesso. Se nell'istruzione CALL COINC si dà un valore non nullo alla tolleranza, si può ottenere la coincidenza anche quando gli sprite sono solo parzialmente sovrapposti, senza che coincidano i punti caratteristici che individuano le loro posizioni. Nel nostro esempio la tolleranza è stata posta uguale a 20 ed è una buona norma che essa sia tanto maggiore quanto più grandi sono gli sprite e quanto maggiore è la loro velocità.

Un automobilista che debba sterzare improvvisamente davanti ad un ostacolo potrà farlo con successo solo se i tempi di reazione lo consentono. Quando il programma prevede il controllo di un oggetto in movimento bisogna tenere conto dei tempi di reazione dell'interprete BASIC e con la pratica e l'esperienza si otterranno le migliori soluzioni. Il listato 4 è un esempio di come il BASIC debba prendere d'anticipo le istruzione che riguardano il movimento e come questo possa essere controllato con opportune pause tra una istruzione e l'altra.

Una pallina aggira un ostacolo che si trova lungo la sua traiettoria e l'ostacolo è posto in corrispondenza della colonna 18. Teoricamente la deviazione della pallina dovrebbe avvenire quando questa raggiunge la colonna 128 (in pixel), in realtà, alla linea 210 del programma, si controlla se è stata superata la colonna 120; quegli 8 pixel di scarto permettono al BASIC di eseguire l'istruzione successiva per tempo, senza che la pallina passi sull'ostacolo.

Le istruzioni CALL MOTION alle linee 220-250 sono state separate da cicli FOR-NEXT il cui solo scopo è quello di creare una pausa durante la quale il movimento prosegue automaticamente. Se la veloci-

tà è uguale a 20, per T che varia da 1 a 100, lo sprite percorre una distanza pari a circa 26 pixel mentre, per T che varia da 1 a 200, lo sprite percorre una distanza di 50 pixel. Come si vede la proporzionalità non è rigorosa.

Oltre a questi brevi esempi voglio presentare un gioco completo che, come al solito, riunisce in modo sintetico tutte le considerazioni precedenti (listato 5).

Partecipano due concorrenti ognuno dei quali "aiuta" una gallina ad attraversare un'autostrada. Il concorrente di sinistra ha a disposizione i tasti (1) e (2) rispettivamente per far avanzare la gallina e per fermarla; analogamente il giocatore di destra dispone dei tasti (9) e (0). Si guadagna un punto per ogni gallina che riesce ad attraversare l'autostrada. Si noti come nello schermo siano presenti contemporaneamente 20 sprite e come si ottenga con sufficiente precisione il controllo delle concidenze per determinare se una gallina sia stata investita. ■

```
100 REM MOVIMENTO CIRCOLARE
110 REM *******************
120 REM       STATEMENT
130 REM      CALL LOCATE
140 REM *******************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"0101010101010107")
180 CALL CHAR(130,"00000000000000FF")
190 CALL CHAR(131,"01010101010101FF")
200 CALL CHAR(132,"0101")
210 CALL COLOR(13,9,1)
220 CALL HCHAR(12,1,130,32):: CALL HCHAR(13,1,132,32)
230 CALL VCHAR(1,16,129,24):: CALL HCHAR(12,16,131)
240 XO=124 :: YO=92 :: RAGGIO=60
250 CALL SPRITE(#1,128,2,YO,XO+RAGGIO)
260 ALFA=0
270 ALFA=ALFA+PI/36
280 X=XO+RAGGIO*COS(ALFA):: Y=YO-RAGGIO*SIN(ALFA)
290 CALL LOCATE(#1,Y,X):: GOTO 270
```

Listato 1. *Traiettoria circolare ottenuta mediante la subroutine LOCATE.*

```
100 REM MOVIMENTO CIRCOLARE
110 REM *******************
120 REM       STATEMENT
130 REM      CALL MOTION
140 REM *******************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"0101010101010107")
180 CALL CHAR(130,"00000000000000FF")
190 CALL CHAR(131,"01010101010101FF")
200 CALL CHAR(132,"0101")
210 CALL COLOR(13,9,1)
220 CALL HCHAR(12,1,130,32):: CALL HCHAR(13,1,132,32)
230 CALL VCHAR(1,16,129,24):: CALL HCHAR(12,16,131)
240 V=-20
250 CALL SPRITE(#1,128,2,92,184)
260 ALFA=0
270 ALFA=ALFA+PI/36
280 VX=V*SIN(ALFA):: VY=V*COS(ALFA)
290 CALL MOTION(#1,VY,VX):: GOTO 270
```

Listato 2. *Traiettoria circolare ottenuta mediante la subroutine MOTION.*

```
100 REM   PROVA JOYSTICK
110 REM ******************
120 CALL CLEAR
130 CALL CHAR(128,"0103030303017F7F6767670
73F3F3030708OC0C3C383FFFFE0E0E0E0E0E0FCF
C04")
140 CALL SPRITE(#1,128,2,100,50)
150 CALL SPRITE(#2,128,9,100,200)
160 CALL MAGNIFY(3)
170 CALL JOYST(1,X,Y)
180 CALL MOTION(#1,-10*Y,10*X)
190 CALL JOYST(2,X,Y)
200 CALL MOTION(#2,-10*Y,10*X)
210 CALL COINC(#1,#2,20,CO)
220 IF CO=0 THEN 170
230 CALL DELSPRITE(ALL)
240 DISPLAY AT(12,12)BEEP:"PRESO!"
250 FOR TEMPO=1 TO 1000 :: NEXT TEMPO ::
 GOTO 120
```

Listato 3. *Piccolo gioco per verificare la coincidenza tra due sprite. È necessario il joystick.*

```
100 REM      MOVIMENTO
110 REM *****************
120 REM      STATEMENT
130 REM   CALL POSITION
140 REM *****************
150 CALL CLEAR
160 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
170 CALL CHAR(129,"F")
180 CALL COLOR(13,9,9)
190 CALL VCHAR(10,18,129,6)
200 CALL SPRITE(#1,128,2,96,40,0,30)
210 CALL POSITION(#1,Y,X):: IF X<120 THEN 210
220 CALL MOTION(#1,-20,0):: FOR T=1 TO 150 :: NEXT T
230 CALL MOTION(#1,0,20):: FOR T=1 TO 100 :: NEXT T
240 CALL MOTION(#1,20,0):: FOR T=1 TO 150 :: NEXT T
250 CALL MOTION(#1,0,30)
260 CALL POSITION(#1,Y,X):: IF X<250 THEN 260
270 CALL LOCATE(#1,96,1):: GOTO 210
```

Listato 4. *Un esempio di movimento con aggiornamento dell'ostacolo.*

Listato 5. *Nel gioco delle galline in autostrada sono state applicate le tecniche descritte.*

```
10 REM ****************
20 REM *              *
30 REM *  LE GALLINE  *
40 REM * IN AUTOSTRADA *
50 REM *              *
60 REM ****************
70 REM  SERGIO BORSANI
80 REM    (0436/3036)
90 REM TI-99/4A EXT.BASIC
100 CALL CLEAR
110 CALL CHAR(96,"7FC1CDCDCDCDC17F"&RPT$
("0",16)&"3F20EFEFEFEF203F"&RPT$("0",16)
)
120 CALL CHAR(100,"FF00FFFFFFFF00FF"&RPT
$("0",16)&"FF01FDFDFDFD01FF"&RPT$("0",16
))
130 CALL CHAR(104,"FF80BFBFBFBF80FF"&RPT
```

Seguito listato 5.

```
$("0",16)&"FF00FFFFFFFFF00FF"&RPT$("0",16
))
140 CALL CHAR(108,"FC04F7F7F7F704FC"&RPT
$("0",16)&"FE83B3B3B3B383FE"&RPT$("0",16
))
150 CALL CHAR(112,"007F717171717F00"&RPT
$("0",16)&"00FE16E6E616FE00"&RPT$("0",16
))
160 CALL CHAR(116,"007F686F6F687F00"&RPT
$("0",16)&"00FE8E8E8E8EFE00"&RPT$("0",16
))
170 CALL CHAR(132,"60E1437F7E3C1828"&RPT
$("0",48))
180 CALL CHAR(121,"0000FFFFFFFF")
190 CALL CHAR(122,"0000FF0000FF")
200 CALL CHAR(123,"0000007E")
210 CALL CHAR(128,"F")
220 CALL COLOR(12,16,15)
230 CALL COLOR(13,15,15)
240 FOR RIGA=5 TO 21
250 CALL HCHAR(RIGA,1,128,32)
260 NEXT RIGA
270 CALL HCHAR(5,1,121,32)
280 CALL HCHAR(7,1,123,32)
290 CALL HCHAR(9,1,123,32)
300 CALL HCHAR(11,1,123,32)
310 CALL HCHAR(13,1,122,32)
320 CALL HCHAR(15,1,123,32)
330 CALL HCHAR(17,1,123,32)
340 CALL HCHAR(19,1,123,32)
350 CALL HCHAR(21,1,121,32)
360 CALL SPRITE(#1,132,16,169,72)
370 CALL SPRITE(#2,132,16,169,160)
380 CALL SPRITE(#3,96,9,41,150,0,-8)
390 CALL SPRITE(#4,100,9,41,164,0,-8)
400 FOR A=5 TO 7
410 CALL SPRITE(#A,112,A,57,2^A,0,-10)
420 NEXT A
430 CALL SPRITE(#8,96,13,73,40,0,-12)
440 CALL SPRITE(#9,100,13,73,53,0,-12)
450 CALL SPRITE(#10,112,7,89,50,0,-20)
460 CALL SPRITE(#11,112,11,89,100,0,-20)
470 CALL SPRITE(#12,116,5,105,10,0,18)
480 CALL SPRITE(#13,116,8,105,80,0,18)
490 CALL SPRITE(#14,104,9,121,60,0,12)
500 CALL SPRITE(#15,108,9,121,75,0,12)
510 FOR A=16 TO 18
520 CALL SPRITE(#A,116,A-8+INT(A*A/100),
137,A*(A-12)*2,0,10)
530 NEXT A
540 CALL SPRITE(#19,104,7,153,120,0,8)
550 CALL SPRITE(#20,108,7,153,135,0,8)
560 CALL MAGNIFY(3)
570 DISPLAY AT(3,10)SIZE(3):0
580 DISPLAY AT(3,15)SIZE(3):0
590 CALL KEY(1,K,S)
600 IF S=0 THEN 630
610 IF K=19 THEN CALL MOTION(#1,-5,0)
620 IF K=7 THEN CALL MOTION(#1,0,0)
630 CALL KEY(2,K1,S1)
640 IF S1=0 THEN 670
650 IF K1=9 THEN CALL MOTION(#2,-5,0)
660 IF K1=10 THEN CALL MOTION(#2,0,0)
670 CALL POSITION(#1,X1,Y1)
680 IF X1>100 THEN 820
690 IF X1>68 THEN 780
700 IF X1>36 THEN 740
710 P1=P1+1
720 DISPLAY AT(3,10)BEEP SIZE(3):P1
730 CALL LOCATE(#1,180,72):: GOTO 950
740 FOR A=7 TO 3 STEP -1
750 CALL COINC(#1,#A,8,C)
760 IF C=-1 THEN 930
770 NEXT A :: GOTO 950
780 FOR A=11 TO 8 STEP -1
790 CALL COINC(#1,#A,8,C)
800 IF C=-1 THEN 930
810 NEXT A :: GOTO 950
820 IF X1>168 THEN 950
830 IF X1>132 THEN 880
840 FOR A=15 TO 12 STEP -1
850 CALL COINC(#1,#A,8,C)
860 IF C=-1 THEN 930
870 NEXT A :: GOTO 950
880 FOR A=20 TO 16 STEP -1
890 CALL COINC(#1,#A,8,C)
900 IF C=-1 THEN 930
910 NEXT A
920 GOTO 950
930 CALL SOUND(100,-3,2)
940 CALL LOCATE(#1,169,72)
950 CALL POSITION(#2,X2,Y2)
960 IF X2>100 THEN 1100
970 IF X2>68 THEN 1060
980 IF X2>36 THEN 1020
990 P2=P2+1
1000 DISPLAY AT(3,15)BEEP SIZE(3):P2
1010 CALL LOCATE(#2,180,160):: GOTO 590
1020 FOR A=7 TO 3 STEP -1
1030 CALL COINC(#2,#A,8,D)
1040 IF D=-1 THEN 1200
1050 NEXT A :: GOTO 590
1060 FOR A=11 TO 8 STEP -1
1070 CALL COINC(#2,#A,8,D)
1080 IF D=-1 THEN 1200
1090 NEXT A :: GOTO 590
1100 IF X2>168 THEN 590
1110 IF X2>132 THEN 1160
1120 FOR A=15 TO 12 STEP -1
1130 CALL COINC(#2,#A,8,D)
1140 IF D=-1 THEN 1200
1150 NEXT A :: GOTO 590
1160 FOR A=20 TO 16 STEP -1
1170 CALL COINC(#2,#A,8,D)
1180 IF D=-1 THEN 1200
1190 NEXT A :: GOTO 590
1200 CALL SOUND(100,-3,2)
1210 CALL LOCATE(#2,169,160)
1220 GOTO 590
```

COMMODORE VIC 20 E C 64

La gestione della tastiera

di Alessandro Guida

La conoscenza di come il computer gestisce la tastiera si rivela una inesauribile fonte di idee. Vedremo come approfittare delle caratteristiche del VIC e del C 64 per ottenere una gestione professionale dei tasti di funzione e come simulare la digitazione di linee BASIC da programma.

Per semplificare la descrizione divideremo il problema in due parti: 1 - la gestione della tastiera; 2 - la gestione dei dati forniti da tastiera.

Vi sarete resi conto che il computer svolge alcune operazioni periodicamente, senza alcun vostro comando,come attivare o disattivare il motore del registratore o incrementare l'orologio interno (variabile TI$). Queste operazioni vengono eseguite 60 volte al secondo, ogni volta che viene attivata da un timer interno la linea di Interrupt. Quando il microprocessore riceve un impulso lungo questa linea interrompe le operazioni che ha in corso e passa ad eseguire una routine, detta Routine di Interrupt. Durante questa interruzione oltre alle operazioni accennate prima, viene eseguito il controllo della tastiera.

L'indirizzo di partenza della Routine di Interrupt è contenuto nel vettore $0314, $0315 (788, 789) ed è normalmente $EABF per il VIC e $EA31 per il C 64.

Come abbiamo detto l'interrupt è provocato da un timer che fornisce un impulso ogni sessantesimo di secondo. Esiste, però una locazione che consente di disabilitare le sorgenti di interrupt (possono essere anche altre diverse dal timer).

Questa è $912E (37166) per il VIC e $DC0D (56333) per il C 64. La tabella 1 ci dà i valori necessari per abilitare o disabilitare l'interrupt generato dal timer.

Proviamo, quindi, il programmino 1. Non funzioneranno più né il tasto di stop né la funzione TI$.

Abbiamo quindi stabilito con sicurezza che la routine di interrupt esegue anche la lettura della tastiera. In particolare ciò viene realizzato dalla KEYBOARD SCANNING ROUTINE memorizzata a partire dalla locazione $EB1F sul VIC e $EA87 sul C 64.

La routine di scansione della tastiera.
La tastiera del VIC (e del C 64) è organizzata come una matrice di 8 x 8 tasti. Fa eccezione solo il tasto di RESTORE che è gestito a parte.

Le otto colonne sono altrettanti bit della porta A di un VIA (Versatile Interface Adapter) mentre le righe fanno capo alla porta B. Ad ogni porta è associata una locazione di memoria.

Il tasto premuto viene identificato attraverso le sue coordinate. Ecco le operazioni svolte dalla routine:
1 Attiva tutti gli 8 bit della porta B e controlla se sulla porta A è presente qualche segnale. Se si vuol dire che un tasto è stato premuto altrimenti la routine viene terminata.
2 Viene testata una riga per volta. Quando una linea è attivata, il computer controlla le 8 colonne e ne ricerca una eventualmente attiva.
Il registro Y del micro viene incrementato ad ogni colonna analizzata.
3 Se nessuna delle colonne risulta attiva, si passa alla riga successiva. Quando questa viene trovata si arresta la procedura di analisi e il numero del tasto contenuto in Y viene salvato nella locazione $CB (203).
4 Viene controllato se il numero del tasto corrisponde allo SHIFT o al simbolo della COMMODORE. In questo caso viene aggiornata la locazione $028D (653), come riportato in tabella 2.
5 Secondo il contenuto di $028D viene scelta una delle 4 tavole contenenti i codici ASCII relativi al numero del tasto premuto. In $F5, $F6 (245, 246) viene conservato l'indirizzo di partenza della tavola scelta. Questa operazione viene svolta in una routine il cui inizio è nel settore $028F, 0290 (655, 656).
6 Utilizzando come puntatore il numero del tasto contenuto in $F5 viene letto all'interno della tavola il codice ASCII corrispondente al tasto.
7 Questo codice viene memorizzato nel buffer di tastiera che inizia in $0277 (631). Viene anche incrementato il contatore dei caratteri nel buffer, $C6 (198). Il numero massimo di caratteri ammessi nel buffer, normalmente 10, è contenuto in $0298 (649).
8 Viene controllato se occorre attivare la funzione di REPEAT. I registri interessati sono i seguenti:

$028A (650)	0 = Repeat solo tasti cursore. 128 = Repeat tutti i tasti.
$028B (651)	Determina il tempo necessario perché inizi la ripetizione.
$028C (652)	Determina la velocità di ripetizione.

9 Termina la routine.

A questo punto possiamo cominciare a trarre alcune conclusioni:

1) Come descritto al passo 3 la locazione $CB contiene sempre il codice del tasto premuto in un certo istante. Abbiamo quindi a disposizione un altro metodo per conoscere durante l'esecuzione di un programma il tasto premuto oltre la nota funzione GET.

La gestione della tastiera

È facile verificare ciò con la seguente linea BASIC:

```
10 PRINT "(Home)" PEEK (203): GOTO 10
```

Date il RUN e vedrete in alto a sinistra il codice del tasto che premete. Se nessun tasto è premuto si legge C 64.

La tabella 3 riporta i codici dei tasti del VIC e del C 64. Il programmino del listato 2 è una routine che utilizza questa possibilità per gestire i tasti di funzione. Chiamando la routine avremo in FK% il numero del tasto funzione premuto, che si potrà utilizzare, ad esempio, con una istruzione di ON FK% GOTO ...

2) Se vogliamo estendere in qualche maniera le funzioni della tastiera è necessario modificare uno dei due vettori visti.

Poiché, come già sappiamo, questi vettori vengono richiamati 60 volte al secondo è necessario disabilitare l'interrupt prima di modificarli, altrimenti si va incontro al blocco del sistema.

Generalmente il vettore che viene modificato è quello contenente l'indirizzo d'inizio della routine d'interrupt. A volte questo può andare bene, ma se intendiamo modificare la routine di gestione tastiera è preferibile cambiare il vettore $028F, $0290.

Ed è ciò che faremo, infatti, per implementare l'uso dei tasti funzione assegnando ad ognuno di essi una stringa.

Memorizzeremo la routine in linguaggio macchina al top della memoria lasciando anche lo spazio per memorizzare le stringhe in modo che non vengano toccate dai comandi CLR e NEW.

Il listato 3 riporta la versione disassemblata per il VIC. Il principio è esattamente uguale nel caso del C 64, per il quale cambiano solo qualche indirizzo e i codici dei tasti. Il disassemblato si spiega da sé. Il listato 4 contiene il modulo in BASIC da caricare e far girare per attivare le funzioni.

Quindi, la pressione di uno dei tasti di funzione fornisce sul video la stringa ad esso associata compreso il carattere di RETURN se vi era stato inserito. Va notato che nell'impostare le stringhe al posto di RETURN bisogna battere la freccia a sinistra (il primo tasto in alto a sinistra).

Per disabilitare l'associazione stringhe-tasti funzione premere contemporaneamente STOP-RESTORE. Per ripristinare il tutto digitare POKE55,0:CLR:SYS (PEEK(251)★256).

Le stringhe sono memorizzate negli ultimi 128 byte di memoria, riservando 16 byte per stringa. La fine di ognuna di esse è segnalata da un byte uguale a zero.

Il programma, nel listato 5, assegna ai tasti funzione le stringhe contenute nelle variabili KE$ (1) ... KE$ (8), e può essere inserito, anche, come subroutine all'interno di un'altro programma.

Bene, per questo mese abbiamo terminato. Nel prossimo numero di **Personal Software** completeremo il discorso. Ci saranno anche due programmi molto interessanti. Buon lavoro. ■

	VIC	64
DISABILITAZ.	POKE 37166,127	POKE 56333,3
ABILITAZIONE	POKE 37166,192	POKE 56333,131

Tabella 1. *Valori necessari per abilitare o disabilitare il timer che fornisce il segnale di interrupt al microprocessore del VIC o C 64.*

TASTO	LOCAZ. $028D
SHIFT	1
COMMODORE	2
SHIFT+COMM.	3
CTRL	4

Tabella 2. *Contenuto della locazione $028D (653) in funzione del tasto premuto.*

	VIC	64		VIC	64		VIC	64
0	1	DEL	22	;	T	44	K	.
1	3	RETURN	23	CRSR R	X	45	:	:
2	5	CRSR R	24	STOP	7	46	=	@
3	7	F7	25		Y	47	F3	,
4	9	F1	26	X	G	48	Q	&
5	+	F3	27	V	8	49	E	*
6	&	F5	28	N	B	50	T	;
7	DEL	CRSR D	29	,	H	51	U	HOME
8	←	3	30	/	U	52	O	
9	W	W	31	CRSR D	V	53	@	=
10	R	A	32	SPACE	9	54	↑	↑
11	Y	4	33	Z	I	55	F5	/
12	I	Z	34	C	J	56	2	1
13	P	S	35	B	0	57	4	←
14	*	E	36	M	M	58	6	
15	RETURN		37	.	K	59	8	2
16		5	38		O	60	0	SPACE
17	A	R	39	F1	N	61	-	
18	D	D	40		+	62	HOME	Q
19	G	6	41	S	P	63	F7	STOP
20	J	C	42	F	L			
21	L	F	43	H	-			

Tabella 3. *Codici tasti letti nella locazione $C5.*

```
10 PRINT"■■■■■■■INTERRUPT ABILITATO"
20 FORI=0TO1000:PRINT"■"TI$,I:GETA$:PRINTA$
30 NEXTI
40 PRINT"■■■■■■■INTERRUPT DISABILITATO"
50 POKE37166,127
60 FORI=0TO1000:PRINT"■"TI$,I:GETA$:PRINTA$
70 NEXTI
80 POKE37166,192
90 GOTO10

PER IL 64 SOSTITUIRE:
50 POKE56333,3
80 POKE56333,131
```

Listato 1. *Prova della disabilitazione dell'interrupt generato dal timer.*

La gestione della tastiera

```
10 REM
20 REM ROUTINE PER ANALIZZARE I TASTI FUNZIONE
30 REM
40 REM
50 REM RESTITUISCE UN NUMERO DA 1 A 8 NELLA
60 REM
70 REM VARIABILE FK%
80 REM
90 REM
100 KEY=PEEK(197): REM LEGGE IL TASTO PREMUTO
110 SH=PEEK(653)AND1: REM 1=SHIFT PREMUTO
120 REM CALCOLA TASTO
130 FK%=-(KEY=39)-3*(KEY=47)-5*(KEY=55)-7*(KEY=63)
140 IF FK%>0 THEN FK%=FK%+SH
150 RETURN

PER IL 64 CAMBIARE:
FK%=-(KEY=4)-3*(KEY=5)-5*(KEY=6)-7*(KEY=3)
```

Listato 2. *Routine per ottenere il numero di tasto funzione premuto.*

Listato 3. *Disassemblato della routine di abilitazione dei tasti funzione.*

```
1D00  78          SEI          ; 'Routine di abilitazione tasti
1D01  18          CLC          ; 'funzione
1D02  A5 37       LDA $37      ; '-----------------------------------
1D04  69 2E       ADC &2E      ; 'Memorizza in $028F,0290 l'indirizzo
1D06  8D 8F 02    STA $028F    ; 'di partenza della routine.
1D09  A5 38       LDA $38      ;
1D0B  69 00       ADC &00      ;
1D0D  8D 90 02    STA $0290    ;
1D10  A5 37       LDA $37      ; 'In $FB,$FC l'indirizzo dei codici
1D12  69 2A       ADC &2A      ; 'dei tasti.
1D14  85 FB       STA $FB      ;
1D16  A5 38       LDA $38      ;
1D18  69 00       ADC &00      ;
1D1A  85 FC       STA $FC      ;
1D1C  A5 37       LDA $37      ; 'In $FD,FE l'indirizzo d'inizio delle
1D1E  69 80       ADC &80      ; 'stringhe associate ai tasti funzione.
1D20  85 FD       STA $FD      ;
1D22  A5 38       LDA $38      ;
1D24  69 00       ADC &00      ;
1D26  85 FE       STA $FE      ;
1D28  58          CLI          ;
1D29  60          RTS          ; '-----------------------------------
1D2A  27 2F 37 3F              ; 'Codici tasti funzione
1D2E  A5 CB       LDA $CB      ; '-----------------------------------
1D30  A0 03       LDY &03      ; 'Controllo se il codice del tasto
1D32  D1 FB       CMP ($FB),Y  ; 'premuto corrisponde ad un tasto
1D34  F0 06       BEQ $1D3C    ; 'di funzione. Nel registro Y vi e'
1D36  88          DEY          ; 'il numero corrispondente da 0 a 3.
1D37  10 F9       BPL $1D32    ; '----------------------------------
1D39  4C DC EB    JMP $EBDC    ; 'Se no salta alla routine originale.
                               ; '----------------------------------
1D3C  C5 C5       CMP $C5      ; 'Se e' lo stesso tasto della volta
1D3E  F0 2C       BEQ $1D6C    ; 'precedente abbandona la routine.
                               ; '----------------------------------
1D40  85 C5       STA $C5      ; 'Lo memorizza per il prossimo contr.
                               ; '----------------------------------
1D42  AD 8D 02    LDA $028D    ; 'Mascheramento dei bit dei tasti
1D45  29 01       AND &01      ; 'Commodore e Ctrl.
1D47  8D 80 02    STA $028D    ; 'Memorizza lo stato dello SHIFT
                               ; '----------------------------------
1D4A  98          TYA          ; 'Riprende il numero del tasto.
1D4B  0A          ASL          ; 'Lo moltiplica per 2.
1D4C  18          CLC          ; 'Vi somma lo shift (1 se e' premuto)
1D4D  6D 8D 02    ADC $028D    ; 'ottenendo il num. del tasto da 0 a 7
                               ; '-----------------------------------
1D50  0A          ASL          ;
1D51  0A          ASL          ; 'Moltiplica il numero del tasto
```

Per 'lavorare' al meglio con il Pet e l'M20

Paolo e Carlo Pascolo

IL BASIC DEL PET E DELL'M20



Il personal computer rappresenta oggigiorno, oltre che un valido aiuto nel lavoro, anche un'irresistibile tentazione. Può capitare, così, che qualcuno si trovi a disporre di un Commodore o di un M 20 Olivetti senza conoscerne appieno il linguaggio e le possibilità. Questo volume vuol rappresentare proprio un prezioso supporto per chi debba, o voglia imparare a programmare in Basic su questi strumenti di lavoro, gioco o studio: comandi, istruzioni, informazioni, consigli... fino a diventare davvero 'padroni' di due dei più diffusi Personal Computer.

226 pagine. Lire 16.000
Codice 336.D

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista

GRUPPO EDITORIALE JACKSON

Per non mandare in tilt il vostro 'cervello'

Rodnay Zaks

PROIBITO!

O come aver cura di un computer



In quanti modi si può rovinare un computer, grande o personal che sia? L'autore di questo volume ne elenca molti: alcuni dovuti a sbadataggine, altri a troppa confidenza con il mezzo, altri ancora a scarsa conoscenza dei suoi meccanismi e della loro estrema vulnerabilità. C'è, anche, un'intera parte dedicata ai sabotaggi da calcolatore: furti, spionaggio industriale, distruzione delle informazioni... Insomma un libro curioso, ma prezioso, per vivere per anni, senza problemi, insieme al proprio amico 'cervello' elettronico.

198 pagine. Lire 14.000 Codice 333 D

GRUPPO EDITORIALE JACKSON

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista

La gestione della tastiera

Seguito listato 3.

```
1D52  0A          ASL         ; 'per 16, ottenendo il puntatore alla
1D53  0A          ASL         ; 'stringa corrispondente.
1D54  A8          TAY         ; 'Conserva il puntatore nel registro Y.
                              ; '------------------------------------
                              ; 'ROUTINE PER COPIARE LA STRINGA NEL
                              ; 'BUFFER DI TASTIERA.
1D55  A2 00       LDX &00     ; 'X=puntatore nel buffer tast.
1D57  B1 FD       LDA ($FD),Y ; 'Carica il carattere Y della stringa.
1D59  F0 0F       BEQ $1D6A   ; 'Se = 0 termina.
1D5B  C9 5F       CMP &5F     ; 'Confronta col carattere freccia.
1D5D  D0 02       BNE $1D61   ; 'Se e' uguale vi sostituisce il
1D5F  A9 0D       LDA &0D     ; 'carattere di RETURN (&0D).
1D61  90 77 02    STA $0277,X ; 'Lo memorizza nel buffer.
1D64  C8          INY         ; 'Incrementa i due puntatori.
1D65  E8          INX         ;
1D66  E0 0A       CPX &0A     ; 'Controlla se e' il decimo carattere.
1D68  D0 ED       BNE $1D57   ; 'Se no torna a leggere il seguente.
1D6A  86 C6       STX $C6     ; 'Memorizza il numero dei caratteri.
1D6C  4C D6 EB    JMP $EB42   ; 'Termina questa routine e quella di
                              ; 'gestione tastiera, tornando alla
                              ; 'routine di interrupt.
```

```
10 POKE56,PEEK(56)-1:POKE55,0:CLR
20 IN=PEEK(56)*256
30 FORI=0TO110:READA:POKEIN+I,A:NEXT
40 FORI=111TO255:POKEIN+I,0:NEXT
50 SYS(PEEK(56)*256):END
1000 DATA120,024,165,055,105,046,141,143,002,165
1010 DATA056,105,000,141,144,002,165,055,105,042
1020 DATA133,251,165,056,105,000,133,252,165,055
1030 DATA105,128,133,253,165,056,105,000,133,254
1040 DATA088,096,039,047,055,063,165,203,160,003
1050 DATA209,251,240,006,136,016,249,076,220,235
1060 DATA197,197,240,044,133,197,173,141,002,041
1070 DATA001,141,141,002,152,010,024,109,141,002
1080 DATA010,010,010,010,168,162,000,177,253,240
1090 DATA015,201,095,208,002,169,013,157,119,002
1100 DATA200,232,224,010,208,237,134,198,076,214,235


MODIFICHE PER IL 64:
1040 DATA088,096,004,005,006,003,165,203,160,003
1050 DATA209,251,240,006,136,016,249,076,072,235
1100 DATA200,232,224,010,208,237,134,198,076,066,235
```

Listato 4. *Modulo BASIC per attivare i tasti funzione. Permette di associare un testo o un comando ad ogni tasto funzione.*

```
10. FORI=0TO7:READ KE$(I):NEXT
20 DATALIST←,GOSUB,RUN←,,LOAD←,GOTO,SAVE,FORI=0TO
90 REM
94 REM ROUTINE PER MEMORIZZARE IL CONTENUTO
96 REM DELL'ARRAY KE$()
98 REM
100 FORI=0TO7
110 LUNG=LEN(KE$(I)):IFLUNG>9THENLUNG=9
120 IFLUNG=0THEN160
130 FORJ=1TOLUNG
140 POKE(PEEK(252)*256+127+I*16+J),ASC(MID$(KE$(I),J,1))
150 NEXTJ
160 POKE(PEEK(252)*256+128+I*16+LUNG),0
170 NEXTI
```

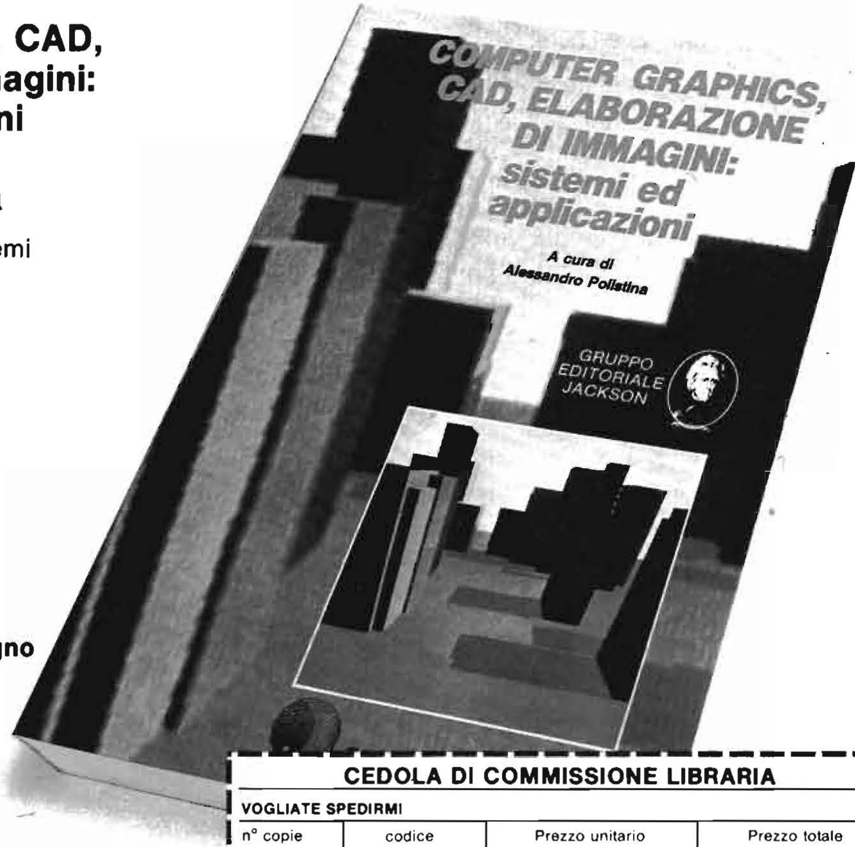
Listato 5. *Routine per assegnare ai tasti funzione le stringhe contenute nel vettore KE$(). Il carattere "←" immette un return nel testo.*

Informatica

Esperti a confronto su attualità e prospettive della Computer Grafica

Computer Graphics, CAD, elaborazione di immagini: sistemi e applicazioni

A cura di
Alessandro Polistina



Linguaggi e algoritmi, sistemi grafici, CAD/CAM, didattica e formazione professionale, Computer Graphics e Editoria, modellazione di solidi, CAD in architettura, CAD meccanico, acquisizione e elaborazione di immagini, elaborazione di immagini e scienze biomediche, cartografia e pianificazione editoriale, immagini sintetiche per la televisione....

Tutti gli Atti del 3° Convegno Nazionale AICOGRAPHICS riuniti in un solo volume a disposizione di operatori, sperimentatori, appassionati. 512 pagine, numerosissimi schemi, un'Appendice con 33 illustrazioni a colori.

Lire 45.000
Codice **529C**

SCONTO 20% AGLI ABBONATI
FINO AL 28-2-'84

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	529C	L. 45.000	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione.

☐ Allego assegno della Banca ______

n° ______

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome ______
Cognome ______
Via ______
Cap ______ Città ______ Prov. ______
Data ______ Firma ______

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. ______

nuovidea

Collisioni per ZX Spectrum

Il lettore Marcello Morchio, anni 15, di Genova, ci invia questa interessante conversione del noto arcade game, pubblicato la prima volta sul n. 3 del Dicembre 1982.

Il programma da me elaborato differisce un po' dall'originale per quanto riguarda le regole e la dinamica del gioco.

La prima differenza è che nella mia versione la macchina comandata dal computer lascia ogni tanto dietro di sé delle mine blu, da evitare perché causano la distruzione della macchina del giocatore; inoltre quando la macchina incontra davanti a sé un carattere grafico, che può essere una mina o la vettura comandata dall'utente, lascia una copia di sé stessa.

L'automobile del computer, infine ha un percorso fisso, perché il suo movimento è determinato dalle linee DATA in fondo al programma (linee 1000 e 1001), che possono essere variate a piacere, ricordando che l'ultimo dato deve essere uno zero per informare il computer che deve essere eseguito un RESTORE per far ripartire il ciclo; naturalmente occorre far sì che il punto di partenza della macchina coincida con quello d'arrivo, per evitare che la vettura tagli le cornici o causi errori di tipo 5 (out of screen) o B (integer out of range).

Un'altra differenza deriva dal fatto che c'è la possibilità, quando si è ripulito uno schermo da tutti i puntini verdi o, come accade più spesso, si è rimasti bloccati fra le mine, di cancellare lo schermo premendo il tasto 0. Tale comando cancella lo schermo, esegue un RESTORE, inizializza i valori delle coordinate delle due macchine e fa ricominciare il gioco togliendo 100 punti dal punteggio.

Questa operazione può essere però eseguita solo tre volte nel corso del gioco; è quindi consigliabile non abusarne.

La macchina si muove con i tasti dei comandi di cursore, (5,6,7,8) e mantiene l'ultima direzione impartita anche se non viene premuto alcun tasto, finché non incontra una barriera o una delle macchine lasciate in giro dal nemico: in questo caso si blocca e vengono sottratti dal punteggio 10 punti per ogni giro di programma, finché non viene mutata la direzione.

Il gioco termina quando la macchina del giocatore viene distrutta dalla macchina del computer o quando incontra una mina. Una curiosità: finito il gioco il computer si pone in attesa che venga premuto un

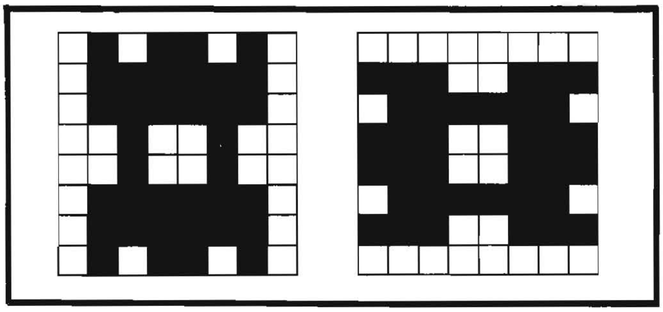


Figura 1. *La macchina del computer nelle sue due posizioni, orizzontale e verticale. All'inizio è verticale (vedi linea 260), ma viene variata a seconda della direzione che si deve prendere (vedi linee da 337 a 340). Si noti come le forme siano simmetriche, per poter immagazzinare in memoria solo due caratteri e non quattro come si dovrebbe volendo rappresentare tutte le direzioni.*

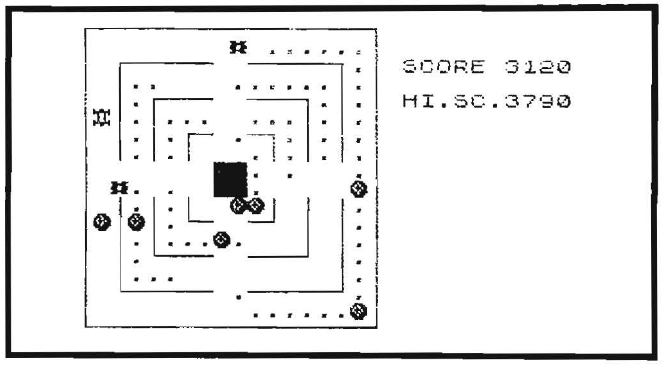


Figura 2. *Una fase del gioco. Si notino le mine e le due macchine uguali (una in alto e l'altra a sinistra) oltre a quella del giocatore. Solo una di queste due si muove, l'altra è una di quelle che il computer lascia per la strada per intralciare il gioco.*

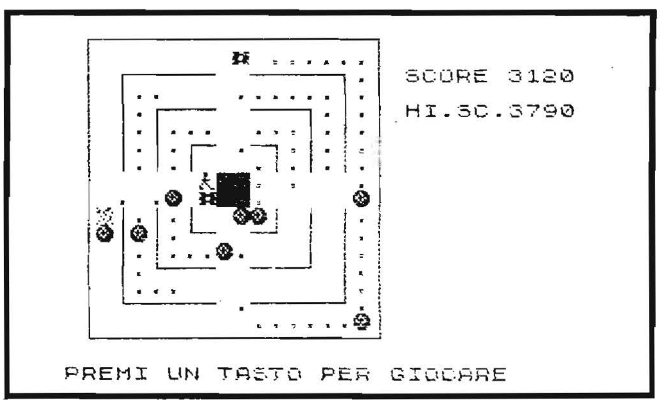


Figura 3. *Fine della partita. Viene visualizzato il messaggio "PREMI UN TASTO PER GIOCARE" e l'omino che balla (in centro, a sinistra del quadrato). È scomparsa la macchina dell'utente, che è stata sostituita dal carattere dell'esplosione nell'impatto con una mina (a sinistra, ultima cornice).*

Collisioni per ZX Spectrum

tasto; durante l'attesa il nostro ZX suonerà la musica di Ufo Robot e il guidatore della macchina avversaria scenderà e si metterà a ballare.

Durante la partita viene costantemente indicato il punteggio corrente e il punteggio massimo ottenuto da quando si gioca.

Buon divertimento a tutti! ■

Note al programma

10-20	Inizializzazione dello schermo (INK, PAPER, BORDER) e della variabile che contiene il punteggio massimo (hi).
100-210	Disegno delle cornici e dei puntini.
220-260	Inizializzazione delle coordinate delle due macchine (x,y per la macchina del giocatore; x1, y1 per quella del computer), dei fattori di incremento di queste (sx, sy, sx1, sy1) e delle stringhe p$ e l$ che contengono le due macchine nella loro ultima posizione (vedi figura 1).
290	Viene letto un dato del file contenente i movimenti della macchina del computer.
295-341	Viene letta la tastiera, aggiornati i valori dei fattori di incremento delle coordinate, aggiornate le coordinate delle due macchine.
342-400	Fase di controllo: viene controllato che la macchina non trapassi una barriera, se è stato mangiato un puntino e se le coordinate delle due macchine coincidono, nel qual caso si passa alla linea 450. Conclusione del loop principale.
450-455	Viene stampato il carattere dell'esplosione e viene simulato uno scoppio con l'istruzione BEEP.
460-490	Loop di attesa con emissione della musica.
1000-1001	Istruzioni DATA contenenti i movimenti della macchina del computer.
1050	Dati contenenti le note e i tempi della canzone Ufo Robot.
8000	Routine per l'inizializzazione dei caratteri grafici.
9000	Salvataggio del programma.

N.B. La corrispondenza dei caratteri grafici è questa: **A**: macchina del computer verticale; **B**: macchina del computer orizzontale; **C**: macchina del giocatore verticale; **D**: macchina del giocatore orizzontale; **I**: mina; **K, L**: posizioni dell'omino che balla; **M**: esplosione.

Listato 1. *Il programma BASIC.*

```
   5 GO SUB 8000
  10 BORDER 7: PAPER 7: INK 0: CLS
  20 LET hi=0
 100 LET p=0: CLS : INK 2: PLOT 12,156: DRAW 0,20-156: DRAW 156-20,0: DRAW 0,156-20: DRAW 20-156,0
 110 PLOT 28,140: DRAW 0,36-140: DRAW 140-36,0: DRAW 0,140-36: DRAW 36-140,0
 120 PLOT 44,124: DRAW 0,52-124: DRAW 124-52,0: DRAW 0,124-52: DRAW 52-124,0
 130 PLOT 60,108: DRAW 0,68-108: DRAW 108-68,0: DRAW 0,108-68: DRAW 68-108,0
 140 PRINT AT 10,9;"██";AT 11,9;"██"
 150 INK 0
 160 PRINT AT 11,2;"       ";AT 11,11;"       "
 170 PRINT AT 10,2;"       ";AT 10,11;"       "
 180 FOR n=3 TO 9: PRINT AT n,9;"  ": NEXT n
 190 FOR n=12 TO 18: PRINT AT n,9;"  ": NEXT n
 195 PRINT AT 4,20;"SCORE ";AT 6,20;"HI.SC.";hi
 200 INK 4: PRINT AT 3,2;"................";AT 18,2;"................";AT 5,4;"............";AT 16,4;"............";AT 7,6;"........";AT 14,6;"........";AT 9,8;"....";AT 12,8;"...."
 210 FOR n=3 TO 18: PRINT AT n,2;".";AT n,17;".": NEXT n: FOR n=5 TO 16: PRINT AT n,4;".";AT n,15;".": NEXT n: FOR n=7 TO 14: PRINT AT n,6;".";AT n,13;".": NEXT n: FOR n=9 TO 12: PRINT AT n,8;".";AT n,11;".": NEXT n
 220 INK 0
 230 LET x=3: LET y=2: LET sx=0: LET sy=1
 240 LET x1=18: LET y1=17
 250 LET p$="▣"
 260 LET l$="▣"
 290 READ dir: IF dir=0 THEN RESTORE : GO TO 290
 291 IF INKEY$="0" THEN RESTORE : GO TO 150: LET p=p-100
 295 IF INKEY$="" THEN GO TO 330
 300 IF INKEY$="8" THEN LET sy=1: LET sx=0: LET p$="▣"
 305 IF INKEY$="5" THEN LET sy=-1: LET sx=0: LET p$="▣"
 310 IF INKEY$="6" THEN LET sx=1: LET sy=0: LET p$="▣"
 320 IF INKEY$="7" THEN LET sx=-1: LET sy=0: LET p$="▣"
 330 LET x=x+sx: LET y=y+sy
 335 IF x=x1 AND y=y1 THEN GO TO 450
 337 IF dir=8 THEN LET sy1=1: LET sx1=0: LET l$="▣"
 338 IF dir=5 THEN LET sy1=-1: LET sx1=0: LET l$="▣"
 339 IF dir=6 THEN LET sx1=1: LET sy1=0: LET l$="▣"
 340 IF dir=7 THEN LET sx1=-1: LET sy1=0: LET l$="▣"
 341 LET x1=x1+sx1: LET y1=y1+sy1
 342 LET ix=0: LET k$=SCREEN$ (x1,y1): IF k$="." THEN LET ix=4
 343 IF x=x1 AND y=y1 THEN GO TO 450
 344 IF ATTR (x,y)=60 THEN LET p=p+50
 345 IF ATTR (x,y)=58 THEN LET x=x-sx: LET y=y-sy: LET p=p-10
 346 IF ATTR (x,y)=57 THEN GO TO 450
```

Collisioni per ZX Spectrum

Seguito listato 1.

```
 350 PRINT AT x,y;p$: IF ATTR (x
-sx,y-sy)<>58 THEN PRINT AT x-sx
,y-sy;" "
 355 BEEP .005,0
 360 PRINT AT 4,23;p
 370 PRINT AT x1,y1; INK 2;l$; I
NK ix;AT x1-sx1,y1-sy1;k$
 380 IF INT (RND*10)=0 THEN PRIN
T AT x1-sx1,y1-sy1; INK 1;"●"
 400 GO TO 290
 450 FOR h=1 TO 3: FOR n=7 TO 0
STEP -1: PRINT AT x-sx,y-sy; INK
 n;"※": BEEP .007,-22: NEXT n: N
EXT h
 455 PRINT AT 21,0; FLASH 1;"PRE
MI UN TASTO PER GIOCARE"
 460 IF p>=hi THEN LET hi=p
 461 LET q=.5/2: LET q1=.5/4
 462 LET ju=1
 465 RESTORE 1050
 470 READ d,n: IF d=1000 THEN GO
 TO 465
 475 BEEP d,n: IF INKEY$<>"" THE
N GO TO 490
 476 IF ju/2=INT (ju/2) THEN PRI
NT AT x1-sx1*2,y1-sy1*2;"🏃": GO
TO 478
 477 PRINT AT x1-sx1*2,y1-sy1*2;
"🚶"
 479 LET ju=ju+1: GO TO 470
 490 RESTORE : GO TO 100
 500 STOP
1000 DATA 7,7,7,7,7,7,7,7,7,7,7,
7,7,7,7,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,6,6,6,6,6,6,6,8,8,6,6,6,6,
```

Seguito listato 1.

```
6,6,8,8,8,8,8,7,8,7,5,5,5,5,7,7,
7,7,8,8,6,6,8,8,8,7,7,7,5,7,7,7,
7,8,8,8,8,8,6,6,6,6,6,6,6,6,6,6,
6,5,5,5,5,5,6,6,5,5,5,5,5,5,5,5
1001 DATA 7,7,7,7,7,7,7,8,8,6,6,
8,8,6,8,8,8,7,7,8,8,8,8,8,8,6,6,
6,6,6,6,6,6
1010 DATA 0
1050 DATA q,4,q,5,q1,7,q,7,q1,7,
q,7,q,9,q,7,q1,4,q1,0,q,-1,q,0,q
1,2,q,2,q1,2,q,2,q,4,q,2,q1,-1,q
1,-5,q,-5,q,-5,q,-3,q*2,-3,q*2,0
,q*2,5,q*2,4,q*5,2
1055 DATA 1000,1000
8000 RESTORE 8100: FOR n=USR "a"
 TO USR "d"+7
8010 READ v: POKE n,v: NEXT n
8020 FOR n=USR "i" TO USR "i"+7
8030 READ v: POKE n,v: NEXT n
8040 FOR n=USR "k" TO USR "m"+7
8050 READ v: POKE n,v: NEXT n
8060 RESTORE : RETURN
8100 DATA 90,126,126,36,36,126,1
26,90,0,231,126,231,231,126,231,
0,153,255,165,36,36,165,255,153,
231,66,126,195,195,126,66,231:
REM caratteri macchine
8120 DATA 60,110,215,187,215,239
,126,60: REM carattere mina
8130 DATA 48,48,32,120,32,80,136
,198,48,48,32,112,160,48,80,80:
REM caratteri omini
8140 DATA 17,138,80,41,144,44,57
,136: REM carattere esplosione
9000 SAVE "collisione" LINE 1
```

SERVIZIO SOFTWARE



PERSONAL SOFTWARE

P.S. propone ai propri lettori i dischi o le cassette dei programmi pubblicati. I programmi, provati e garantiti, sono di immediato utilizzo.

P.S. n°	Programma	Sistema	Prezzo	Codice	Supporto
3	La carta del cielo Collisione	Apple II	30.000	1	Disco
3	Backgammon	TRS-80 Mod. I	25.000	2	Disco
2	Editor/Assembler in BASIC	CBM 3032	40.000	3	Disco
4	Interi in precisione multipla Grafica 3D	Apple II	40.000	4	Disco
4	Gioco del calcio	CBM 3032	25.000	5	Disco
5	Pretty printer Shape table	Apple II	30.000	6	Disco
7	Data base modulare	Apple II	25.000	7	Disco
12-13	Wei-ch'i	CMB 3032	20.000	8	Cassetta
14	Tool-Kit	C 64	35.000	9	Cassetta

Per richiedere i programmi in contrassegno, pagando direttamente al postino la cifra indicata, inviare il seguente tagliando

Spedire in busta chiusa a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Inviatemi i seguenti nastri e/o dischi con i programmi pubblicati su **P.S.**

GRUPPO EDITORIALE JACKSON

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Spese postali (contributo fisso) L. 2.000

TOTALE L. ....

che pagherò al postino alla consegna del pacco.

Cognome ....

Nome ....

Indirizzo ....

CAP ....

Città ....

Firma ....

PICCOLI ANNUNCI

Apple

Vendo interfaccia IEEE-488 per Commodore 64 usata pochissimo a L. 150.000 trattabili. Flavio Stella - Via Grandi, 10 - 20060 Cassina De' Pecchi (MI) - Tel. 02/9521017 (ore serali).

Vendo "Language card" originale (16 K RAM + Integer Basic) per Apple IIe compatibili a L. 155.000, prezzo di listino L. 352.000 + IVA. Regalo un programma a scelta. Tiziano Settimi - Via XXIV Maggio, 30 - 20010 Canegrate (MI) - Tel. 0331/400303.

Vendo per Apple il famoso "The last one" con manuale in italiano a L. 250.000. Sistema operativo Pascal a L. 200.000. Sistema gestionale e magazzino in italiano con manuali a L. 800.000 e centinaia di altri programmi. Mauro Marcon - Via Posati, 10 - 31010 Asolo (TV) - Tel. 0423/55395 (ore pasti).

Vendo/cambio programmi Apple per tutti i gusti. Rivolgersi a Franco Vittor - Via Grabizio, 35 - 34170 Gorizia - Tel. 0481/81254.

Cambio di programmi per Apple specialmente giochi in linguaggio macchina. Attendo vostre liste. Scambio solo su dischetto. Enrico Sturaro - C.so Casale, 416/6 - 10132 Torino - Tel. 011/898756.

Cambio programmi Apple II e IIe. Cesare Giardini - Via Castellana, 39 - 27029 Vigevano (PV) - Tel. 0381/21405.

Vendo Software per Apple II e Spectrum. Vasto assortimento, più di 250 programmi. Prezzi eccezionali (L. 7.000 Spectrum 48 K). Sconti a chi decide di comprare più programmi. Roberto Dal Tio - Via Pianale - 31020 S. Maria Di Feletto (TV) - Tel. 0438/784050.

Cambio/Vendo per Apple programmi di ingegneria e/o utility. Scrivere o telefonare a: Pietro Petrocchi - Via G. Perticari, 5 - 61035 Marotta (PS) - Tel. 0721/96434.

Vendo per Apple II scheda espansione microframe 28 K, compatibile Ramex, con software simulatore di disco Superdos a L. 500.000. Telefonare ore serali. Silvio Valentini - Via G. Malaspina, 9 - 35100 Padova - Tel. 049/691475.

Cerco per Apple IIe: Apple writer IIe - Quick file - PFS IIe con manuali. Ezio Martelletto - Via G. Mameli, 24 - 36100 Vicenza.

Vendo "The last one" versione Apple II completo di manuali in italiano a L. 300.000. Achille Betti - Via Del Brennero, 109 - 55100 Lucca - Tel. 0583/953411 (ore pasti).

Vendo per Apple II: "Language card" originale (16 K RAM + linguaggi) a L. 155.000. Prezzo di listino L. 352.000 + IVA e scheda "EPROM programmer" con disco originale a L. 185.000. Tiziano Settimi - Via XXIV Maggio, 30 - 20010 Canegrate (MI) - Tel. 0331/400303.

Vendo programmi per Apple II su disco e su cassetta. Filippo Salomone - Via P. Giuliani, 5 - 21047 Saronno - Tel. 02/9625998.

Cerco per Apple II, o altri, informazioni per utilizzazione musicale, soprattutto, frequenze, toni, ecc. Rimborso spese postali. Stefano Malagodi, Casella Postale - 44034 Copparo (FE) - Tel. 0532/860196.

Vendo per Apple II "Supertoto 1.0", superprogramma totocalcio inedito, 3 diverse opzioni di selezioni incrociate (N° segni 1 X 2; consecutivi; corr. errori), con output N° colonne utili, sviluppo su monitor o stampante. L. 70.000 con manuale. Rossi Roberto - Via Lario, 26 - 20159 Milano - Tel. 02/6070236.

Scambio/Vendo programmi per Apple II: grafica, compilatori, business, ma in particolare giochi e copiatori. Scrivere o telefonare a: Filippo Buelli - Via Sacconi, 4 - 20052 Monza (MI) - Tel. 039/364922.

Cambio "The last one" (con manuale) versione per Apple II-IIe, con scheda 80 colonne sempre per Apple II. Tiziano Settimi - Via XXIV Maggio, 30 - 20010 Canegrate (MI) Tel. 0331/400303.

Atari

Cerco possessori Atari 400/800 per acquisto/scambio/vendita di programmi originali su disco e cassetta. Luigi Servolini - Via La Spezia, 81 - 00182 Roma - Tel. 06/7581219 - Tel. 384488.

Vendo Atari VCS 2600 con relativi comandi e trasformatore con sette cassette: Brekout, Defender, Combat, Air sea battle, Street racers, Maze craze, Asteroids con relativo impacco. Valore commerciale L. 750.000, vendo il tutto a L. 500.000.
Mauro Rorato - Via dei Tigli, 2/A - 20090 Rodano 1000 pini (MI) - Tel. 9588000.

Vendo Atari 400 modello americano + alimentatore + cartuccia Basic + molti fantastici programmi su nastro. Per informazioni telefonare o scrivere a: Paolo Marcato - Via Cesare Battisti, 3 - 35027 Noventa Padovana (Padova) - Tel. 049/502475.

Compro i n° 1-3-4-5-6-8-9-11-12-13-14 di **Bit** e n° 1-2-4-11-25 di M&P. Telefonare o scrivere a: Carlo Cocciazucca - Via Montesecco, 15 - 65010 Spoltore (PE) - Tel. 085/207466 (ore pasti).

Vendo vera occasione, videogame Atari perfettamente funzionante con cassette (Soccer, Space invaders, Combat) e 2 coppie di pulsanti a L. 250.000. Fabrizio Ceccarelli - Via Empoli, 131 - 47025 - Cesena (FO) - Tel. 0541/331656.

Acquisto/scambio/vendo programmi per computer Atari 400-800 su cassetta e disco. Luigi Servolini - Via La Spezia, 81, 00182 Roma - Tel. 06/7581219 384488.

Commodore

Scambio/compro programmi per CBM-64 Commodore. Gianluigi Peduto - Via Malgrado - 40125 Bologna.

Svendo per avvenuto passaggio a sistema superiore 210 programmi per VIC-20. Utility, matem., giochi tra cui molti in L.M. (Bonzo, Alien, Boss). Svendo a L. 25.000 comprese spese di spedizione e nastri magnetici. Marco Pierbattisti - Via R. Donatelli, 5 - 05100 Terni (TR) - Tel. 0744/418277.

Commodore 64 *cerco* utenti per scambio programmi e formazione gruppo per acquisto programmi in comune. Rispondo a tutti. Comunicare anche recapito telefonico. Gianpiero Piacentini - Via G. Mameli, 115 - 00040 S. Maria Mole (Roma) - Tel. 06/9351150.

Cambio software per VIC 20 da 6-8 K, anche non espanso. Diego Braganti - Via C. Ravizza, 40 - 20149 Milano - Tel. 02/4989786.

Per VIC-20 *cerco* 8 K ROM giochi e utility in ottimo stato, software gestionale che richiede il disco. Inoltre ho 320 prog. da scambiare se inviate vostre liste; invio catalogo di vendita per L. 1.000 (circa 20 pagine). Giorgio Ferrario - Via Adua, 1 - 21052 Busto Arsizio (VA).

Cerco buon prezzo registratore a cassette Commodore CZN per il mio C64. Giorgio Maselli Campagna - Via Macchie, 31/8 - 70057 Palese (BA) - Tel. 320400.

Vendo a ottimi prezzi programmi per VIC-20 di ogni genere. Per richiedere la lista spedire indirizzo più L. 500 in francobolli a: Fernando Benini - Via E. Pazzi, 16 - 48100 Ravenna.

Vendo VIC-20 completo alimentatore, interfaccia, video, espansione 16 K, joystick, unità cassette C2N + manuale + VIC revealed a L. 700.000. Massimiliano Bottacini - Via Roma, 48 - 15010 Rivalta Bormida (AL) - Tel. 0144/72116.

Vendo per VIC-20 interfaccia VCX-1001 (adatta anche a Pet). Scambio inoltre software per VIC. Inviare lista o richieste, rispondo a tutti. Alessandro Bruciamonti - Via Roma, 72 - 27047 S. Maria della Versa (PV) - Tel. 0385/79052.

Vendo per CBM-64 interessantissimi programmi. Molti programmi tecnici (es.: revisione prezzi, cadute di tensione), utility e giochi. Assicuro risposta a tutti. Chiedere la lista allegando L. 300 in francobolli. Claudio Tioni - Via Cividina, 59 - 33035 Martignacco (UD) - Tel. 0432/677627.

Vendo listati per VIC: Star war, Pac-man, Tiro al drago, Labirinto, Donkey kong, Frogger, basket, VIC dietologo, Matematica, Orologio, Bach, Bioritmi, VIC medico, Battaglia navale. Prezzo L. 2.500 cad. Massimo Gusso - Via Felissent, 32 - 31020 Lancenigo (TV) - Tel. 62969.

Vendo/scambio software (specialmente giochi) per Commodore 64 programmi GB/USA. Inviare lista programmi (rispondo a tutti). Roberto Delbello - Via Giarizzole, 7 - 34100 Trieste.

Vendo VIC Expansion Arfon con 7 slot + coperchio ancora imballato a L. 300.000; nuovo L. 375.000 e stampante VIC 1525 30 C/S 80 colonne nuova + 1800 fogli carta a L. 650.000 anzichè L. 785.000. Scrivere per accordi. Walter Della Spora - Via P. Savi, 218 - 55049 Viareggio.

Vendo Software civile C-64 Spectrum, manuali, supertestato, telai grigliati, verifiche, fond. zona sismica, 373, ecc. Giovanni Gaviani - Via Finelli, 3 - 40100 Bologna - Tel. 051/230126.

Vendo CBM 3032 + 4040 Epson MX80FT + 90 dischi con centinaia di progr. gestionali mat, ing. con manuali + tantissimo altro materiale. Il tutto è nuovissimo e lo offro al migliore offerente anche a rate dilazionate. Sergio Sonagere - Via Kennedi, 35 - 33038 S. Daniele Del Fr. (UD) - Tel. 0432/955666.

PICCOLI ANNUNCI

Scambio programmi per VIC-20; inviate cassette o listati. Massima serietà. Sono pronto a ricambiare le vostre lettere. Marco Ceruti - Via Frassinago, 59 - 40132 Bologna - Tel. 585389.

Cambio/vendo programmi per il VIC-20. Ne possiedo circa 400. Richiedere il catalogo con la delucidazione di ogni programma inviando L. 1.000 in francobolli. Enrico Sturaro - C.so Casale, 416/16 - 10132 Torino - Tel. 011/898756.

Vendo computer Commodore CBM 4032, registratore C2N, manuale d'uso, cassette con programmi. Il tutto a L. 1.100.000 trattabili. Fabrizio Corsello - C.so Turati, 82 - 10134 Torino - Tel. 011/599184.

Vendo VIC-20 + registratore C2N + cartuccia scacchi + 3 manuali + 5 programmi a sole L. 400.000 trattabili. Valentino Mosca - Via G. Silla, 8 - 00189 Roma - Tel. 3765394 (ore pasti).

Scrivi o telefona: potremmo sviluppare o cambiare software per il nostro Commodore 64! Insieme, possiamo fare di più!. Ivan Zoratti - Via Cilea, 86 - 20151 Milano - Tel. 02/3533359.

Cerco qualsiasi tipo di programma per CBM-64. Inviare liste con eventuali prezzi. Sono disposto anche a scambiare miei eventuali programmi. Scrivere a: Matteo Ficco - Via S. Giovanni da Verdara, 87 - 35100 Padova - Tel. 049/653482.

Vendo VIC-20 + Super-expander 3 K + 8 K RAM + programmer's AID + registratore C2N + 3 manuali, per passaggio sistema superiore. Luca Lodoletti - Via Solari, 2 - 20144 Milano - Tel. 02/482891 (ore pomeridiane).

Svendo programmi, linguaggi, compilatori e utilità per computer Commodore serie 3000, 4000, 8000 causa passaggio altro sistema. Richiedere elenco dettagliato inviando L. 1.000 a: Antonio Marocco - Via Dell'ingegno, 35 - 34073 Grado (GO).

Cambio/compro/vendo programmi per VIC-20. Massima serietà. Scrivere a: Francesco De Colle - P.le Capolinaro, 11 - 00053 Civitavecchia (Roma) - Tel. 0766/34171.

Vendo VIC-20 + registratore in perfette condizioni a L. 320.000. Vendo inoltre, anche separatamente, vari programmi di alta qualità (scacchi, giochi spaziali, ecc.) a metà prezzo. Giorgio Pietrocola - Via Caneva, 25 - 00159 Roma - Tel. 06/4388795.

Per il VIC-20 *vendo* programmi in linguaggio macchina a favolosi prezzi: Abductor, Amok, Blitz, Guardian e molti altri. So duplicare le cartridge in programmi da 16 K. Roberto Silva - Via L. Cagnola, 20154 Milano - Tel. 02/317228.

Scambio oltre 150 programmi (videogiochi, gestionali, utility) per Commodore-64. Luciano Cuneo - Via E. Lepido, 46 - 00175 Roma - Tel. 06/7491542.

Vendo/scambio "Il libro del Commodore VIC-20", "Guida al personal VIC-20", cassetta originale inglese (7 giochi) per VIC non esp. Prezzi da concordare. Carlo Avino - Via A. Baccarini, 46 - 00179 Roma - Tel. 06/7885291 (ore 18-21,30).

Vendo Commodore 64 con numerosi programmi e manuale in italiano; il tutto garantito nuovo a L. 700.000. Tratto con provincie di Genova, Savona, La Spezia, Massa-Carrara. Roberto Garibaldi - Via Rino Mandoli - 16139 Genova - Tel. 010/892853 (ore pasti).

Vendo VIC-64 nuovo (causa doppio regalo) a L. 480.000. Vendo inoltre VIC-20 a L. 190.000 (ottimo stato), 16 Kbyte L. 110.000, tool-kit (prog. AID) L. 25.000, linguaggio macchina (VIC-MON) L. 25.000, VIC-REL L. 65.000. Software gratuito. Aldo Stracchi - V.le Europa, 170 - 39100 Bolzano (Tel. 0471/931448.

Vendo per VIC-20 "Agenda", su cassetta, per creazione archivio indirizzi, o altri tipi definiti dall'utente. Minima espansione 3 K - min. 60 indirizzi; max ca. 2000 con 32 K. Il numero dipende dall'archivio. Prezzo L. 10.000 + spese postali. Vincenzo Carrone - Via Pascoli, 67 - Campobasso - Tel. 0874/91995.

Vendo/cambio software per VIC-20. Prezzi incredibili!! Bonzo, Sub chase, Frogger, Crazy Kong, Asteroids sono solo alcuni esempi. Richiedi la lista allegando L. 400 per spese di spedizione a: Gregorio Lena - V.le Silvani, 3/2 - 40122 Bologna - Tel. 551178.

Cambio programmi in cassetta per Commodore 64. Inviare liste. Scrivere a: Paolo Di Mauro - Via Bertieri, 1 - 20146 Milano - Tel. 02/471803.

Per VIC-20 *vendo* cassetta con 20 fantastici giochi di animazione in tempo reale, a colori e sonori originali inglesi con istruzioni in italiano, funzionanti con memoria base, tutto a L. 30.000. Claudio Giovanelli - Via Ripamonti, 194 - 20141 Milano - Tel. 02/536926.

Cambio/vendo programmi per Commodore 64, giochi, grafica, utilità e di altro tipo. Eliseo Bergamo - Via Rocche, 7 - 36077 Altavilla (VC) - Tel. 980840 (ore serali).

Vendo/cambio ottimi programmi LM e Basic per VIC-20. Anche ottimi inediti e utility. Scrivete per ricevere la lista gratuita. Federico Gurrieri - Via U. Foscolo, 14 - 50124 Firenze - Tel. 055/700635.

VIC-20 + interfaccia registratore + oltre 30 programmi + 2 libri sul VIC-20 a sole L. 300.000. Espansione 16 K a L. 160.000; tutto usato pochissimo a L. 450.000. Nicola De Vita - Via Val D'Ala, 20 - 00141 Roma - Tel. 8102121.

Svendo per CBM-64 causa rottura irreparabile, completa raccolta di tutti gli articoli e programmi apparsi finora in Italia; manuali d'uso perfettamente tradotti ed interfaccia per registratore. Fabrizio Besetti - Via G. Verdi - 24040 Canonica d' Adda (BG) - Tel. 035/883107 (ore ufficio).

Vendo molti programmi a prezzi modici; sei un VIC-utente alle prime armi? Rivolgiti a me: dispongo di molti programmi dimostrativi per capire concetti fondamentali del BASIC del VIC-20 a basso prezzo e offro lezioni di BASIC per corrispondenza. Fabio Siani - Via Buonarroti, 19 - 20149 Milano - Tel. 02/4694089.

Sinclair

ZX-81 *cerco* possessori per scambio programmi su listato o cassetta per 1 K e 16 K RAM. Scrivere per accordi spedendo eventualmente una lista a: Giovanni Pietrolonardo - Via G. Pelosi, 41 - 00143 Roma.

Vendo software originale importato direttamente dalla Gran Bretagna per Computer Spectrum. Allegare francobollo per risposta. Antonio mauriello - C.so Secondigliano, 209 - 80144 Napoli.

Vendo per Computer ZX-81 fantastica cassetta C-60 contenente 40 programmi max. da 1 K, 20-Game, 20-Utilitye a L. 5.000 più L. 2.000 spese postali. Per ordinarla scrivere o telefonare a: Pierangelo Patrizi - Via del Mare, 47 - 73100 Lecce - Tel. 0832/52891 (ore pasti).

Cerco possessori ZX Spectrum per scambio software alla pari e notizie sullo stesso. Scrivere accludendo propria lista programmi e bollo per la risposta. Luigi Ballestin - Via Martiri della Libertà, 367/11 - 18038 Sanremo (IM).

Vendo Spectrum 48 K completo + manuale in italiano tutto in ottimo stato con imballaggio originale (5 mesi di vita) a L. 430.000. Giuseppe Scavo - Via G. Ferraris, 9 - 28100 Novara - Tel. 0321/454679 (ore pasti).

Softmatematico per Spectrum o ZX-81, cassetta con 8 programmi: integrali, equazioni differenziali, sistemi lineari ecc. solo L. 15.000. Scrivere o telefonare a: Paolo Biagioni - Via Lungo L'Affrico, 84 - 50137 Firenze - Tel. 055/664476.

ZX Spectrum 16/48 Kbyte *vendo* cassetta con due giochi (Labirinto 3D - Poker) Basic autocompilati L.15.000. Listati L. 5.000 cad. Spedizione contrassegno. Sebastiano Trusso - Via Roma, 291/A - 98051 Barcellona P.G. (ME) - Tel. 0190/9723167.

Vendo/cambio programmi Spectrum a prezzi stracciati. Cambio anche i programmi con altri possessori di Spectrum per aumentare il numero di programmi. Telefonare o inviare elenco a: Alessandro Carbonara - Via Faenza, 159 - Triggiano (BA) - Tel. 080/681928.

Vendo Sinclair ZX-81 + espansione 16 K, alimentatore, cavetti e manuale inglese e italiano. L. 200.000 completo. Regalo all'acquirente un libro con programmi + 1 cassetta. Alessandro Pelati - Via A. Ciseri, 32 - 50142 Firenze - Tel. 055/780642.

Vendo ZX-81 causa passaggio sistema superiore + espansione 16 K + tastiera speciale + 5 cassette programmi 16 K + libri e manuali. Valore totale L. 450.000 vendo L. 350.000 trattabili o scambio con ZX Spectrum 48 o 16 K. Fabrizio Vita - V.le Monza, 26 - 20127 Milano - Tel. 02/2850136.

Cerco possessori di ZX Spectrum, preferibilmente zona Rovigo, per scambio-compra-vendita software. Scrivere o telefonare: Gabriele Formaggio - Via Dante Gallani, 23/a - 45100 Rovigo - Tel. 35726.

Scambio programmi per ZX-81 16 K in cassetta o listato; inoltre scambio alimentatore ZX 0,7 Amps con inverse video. Infine scambio 5 rotoli di carta termica Sinclair con tastiera applicabile su ZX. Paolo Ballocci - Via S. Gottardo, 75 - 20052 Monza (MI) - Tel. 039/367709.

Vendo per Sinclair ZX Spectrum set di programmi finanziari, statistici e scientifici completo di esauriente guida all'utilizzo ed allegati esplicativi. Prezzo incredibilmente basso. Per informazioni scrivere a: Giosuè Baiano - Via C. Battisti, 11 tr. priv. - 80059 - Torre del Greco (NA).

Vendo eccezionali programmi su cassetta per ZX Spectrum nella versione 16 o 48 K a L. 10.000. Per informazioni rivolgersi a: Michele Stagno - Via C. Pompea, 285 - 98100 Messina - Tel. 45018.

PICCOLI ANNUNCI

Vendo/cambio software acquistato in GB. Per Spectrum 48 K: Football manager, Warlord, Jackpot, Battle of Britains, PSST, Flight simulation. Tratto con zona Genova. 1 cassetta L. 15.000; 6 L. 75.000. Giorgio Vanni - Via Gaulli, 7 - 16143 Genova - Tel. 010/512248 (ore serali).

Vendo Game program oroginali inglesi per ZX Spectrum a prezzi molto buoni. Paolo Fiorino - Via Giambellino, 102 - 20146 Milano - Tel. 02/4238712.

Per ZX-Spectrum *cerco* programma "ZX slow loader". Chi lo possedesse mi spedisca pure la sua lista di programmi per eventuali altri scambi o acquisti. Polano Alberto - Via D. Chiesa, 14 - 33038 S. Daniele (UD).

Vendo ZX Spectrum 48 K nuovissimo a L. 350.000. Giancarlo Mariani - V.le Brianza, 72 - 20036 Meda (MI) - Tel. 0362/72565 (ore pasti).

Vendo/scambio software per ZX Spectrum 16/48 K. Oltre 200 progammi disponibili! Inviare catalogo. Bruno Rota - Via Passo di Brizio, 6 - 20148 Milano - Tel. 02/4082437.

Vendo (eventualmente scambio) i migliori programmi per ZX Spectrum a prezzi incredibili: Chequered flag, Hobbit, Pascal, compilatore Basic e altri 48 K a L. 10.000; Jet Pac e altri 16 K a L. 5.000. Spedire L. 400 in francobolli a: Francesco Zanichelli - Via Traversetolo, 192 - 43030 Porporano (PR) - Tel. 0521/641165.

Cedo software per ZX Spectrum 16 e 48 K. Ho anche listati per altri computer. Telefonare oppure scrivere a: Gianmauro Dell'Olio - Via Marchese di Montrone, 60 - 70122 Bari - Tel. 080/219840.

Vendo per passaggio sistema superiore Sinclair ZX-81 + espansione 16 K RAM + alimentatore + manuale italiano + libro "66 programmi per ZX-81" + cassetta database il tutto agosto '83 a L. 220.000 trattabili. Walter Bianchi - Via Casoni, 84 - 31021 Mogliano Veneto (TV) - Tel. 041/454735.

Finalmente software per Spectrum su cassetta in italiano. Programmi originali o comunque tradotti e comprensibili. Oltre 80 titoli a L. 4.000 e L. 5.000 rispettivamente per 16 o 48 K. Richiedere lista. Gianfranco Posterli - Via L. Ariosto, 123 - 20099 Sesto S. Giovanni (MI) - Tel. 02/2480163.

Spectrum *scambio/vendo* programmi gioco e/o utilità; rapida risposta; elenco gratuito. Scrivere o telefonare a: Antonio Sfriso - Via Salomone, 7 - 30173 Mestre (VE) - Tel. 041/972887(ore cena).

Vendo computer ZX-81 Sinclair usato pochissimo, ancora in garanzia. Insieme al computer riceverete anche il libro istruzioni in inglese e italiano più l'alimentatore. Tutto a L. 90.000. Mario Porchera - Via Lambro, 6 - 20089 Rozzano (MI) - Tel. 8257086.

Vendo Sinclair ZX-81 completo di alimentatore e cavetti, espansione di memoria 16 K RAM, manuale in italiano e numerosi giochi su cassetta. Tutto a L. 200.000 trattabili. Vendo anche separatamente. Roberto Lopez - Via C. Troya, 2 - 20100 Milano - Tel. 02/425908.

Vendo ZX80 + 8 K ROM + 16 K RAM + slow + registratore (il tutto in contenitore) + tastiera esterna + software su nastro + manuali L. 300.000 intrattabili. Carlo Cecchi - Via Monferrato, 15 - 20144 Milano - Tel. 02/4959020.

Offro ZX81 16 K + alimentatore, manuale e cavetti + vasto software package, tra cui: scramble, Meteorites, MA20GS - Breakout a L. 300.000 trattabili. Cesare Giavani - Via Troubetzkoy, 82 - 28058 Verbania-Suna (NO) - Tel. 0323/504182 (ore pasti).

Texas

Cerco/scambio programmi per il TI 99/4A (solo su cassetta). Romano Perico - Via Geroni, 2 - 24025 Cazzaniga (BG) - Tel. 035/711993.

Cerco piccoli programmi in Assembler TI-99/4A per sopperire alla poca chiarezza del manuale. Rosario Velardi - Via Mare di Bering, 40 - 00122 Ostia Lido (Roma) - Tel. 5684912.

Vendo per TI-99/4A programma, scritto in TI-Basic, grafico, a colori, sonoro, della "Battaglia navale". Costo listing L. 20.000: prezzo stracciatissimo. Spedizione contrassegno. Valentino Ricci - P.zza Spirito Santo, 32 - 65100 Pescara.

Vendo TI-99 + cavo registrazione + manuali + extended Basic + teach your self XBASIC + the attack + cassette di giochi e programmi di varia utilità. Il tutto a L. 500.000 trattabili. Michele Cei - Via Colombo, 7 - 27100 Pavia - Tel. 0382/27797.

Vendo TI-99/4A completo L. 310.000, extended Basic con manuale L. 180.000, mini memoria + cassetta Assembler L. 210.000, manuale Assembler 450 pagine L. 20.000, coppia joystick L. 40.000. Con garanzie. Alessandro Pasciuto - Via Nervesa Della Battaglia, 7 - 80124 Napoli - Tel. 081/618624.

Vendo TI-99/4A + TI invaders + cavo per due registratori + manuali italiani. Garanzie da spedire. L. 400.000 trattabili. Regalo inoltre un favoloso programma (in vendita a L. 15.000 separatamente) di analisi del campionato di calcio. Schede squadre, 29 classifiche tra le quali una basata su 20 fattori. Per sistemisti e appassionati. Davide Rolando - Via B. Ottaviano, 6/6 - 17100 Savona - Tel. 019/26949.

Vendo per TI-99/4A cassetta "Adventure" dal titolo il "Conte" necessario modulo "Adventure". Disposto successivamente a fornire indicazioni per il funzionamento e la risoluzione del gioco. Stato discreto. L. 30.000. Fabio Ravanelli - V.le Kennedy, 105 - 28100 Novara - Tel. 0321/451953.

Vendo TI-99/4A + registratore della Texas + cavetto di collegamento a L. 400.000. Tutto in garanzia. Giovanni Amico - Via Houel, 19 - 90138 Palermo - Tel. 335226.

Vendo cassette per TI-99: Wumpus, Videogames 1, Parsec, Pirate adventure + modulo 555, Il conte, Il castello del woodoo. L. 30.000 cad. (la metà del prezzo reale!). In blocco L. 150.000. Fabio Ravanelli - V.le Kennedy, 105 - 28100 Novara - Tel. 0321/451953.

Cambio/vendo programmi per TI-99, anche originali Texas. Cerco inoltre utenti di questo computer di Firenze per scambio idee, ecc. Alberto Bemporad - Via L. Settembrini, 20 - 50133 Firenze - Tel. 055/470620.

Varie

Vendo computer MPFII, 164 RAM, Applesofit compatibile, come nuovo + tastiera esterna + alimentatore + interfacce recorder e TV + manuali Basic e LM + molti programmi, listati, ecc. Prezzo eccezionale. Umberto Torrini - Via Bolognese, 57 - 50139 Firenze - Tel. 055/474836-265033.

Vendo Micro Z80 N.E. LX 380, 381, 382, 383, 384, 385, 386, 387, 388; tutto funzionante con mobile contenitore, registratore, Basic 5,5 K, cassette programmi e manuali di istruzione L. 600.000 (nuovo). Carlo Vincenzi - Via Resistenza. 26 - 41033 Concordia S/S Modena.

Vendo HP-85 con 32 K RAM stampante 80 ÷ 132 colonne mod. HP-82905A interfaccia HP-1 B ROM plotter/printer ROM matrix - advanced programming (valore di listino L. 9.640.000). Tutto per L. 6.000.000 trattabili. Barbara Mereu - Via Alghero, 45 - 09100 Cagliari - Tel. 070/652652.

Vendo per Micro Z-80 NE - configurato con scheda grafica (LX529) e almeno 1 driver - dischetto 5" completamente riempito da 12 programmi nuovi di gioco e grafici. Il tutto a L. 20.000 + spese postali in contrassegno. Federico Venier - Via Venezia, 120 - 33170 Pordenone - Tel. 0434/42500.

Vendo EPROM con extendet-Basic per Acorn Atom da inserire nello zoccolo per le utilitie. Aggiungere 21 istruzioni tra le quali: READ, DATA, RESTORE, INKEY$, PRINT AT, TAB, SCREEN, ecc. Massimo Magnani - Via Tibullo, 10 - 47044 Igea Marina (FO) - Tel. 0541/630470.

Vendo causa passaggio a sistema superiore HP-4 ICV + stampante HP-82 143A + modulo elettronica + manuali. Il tutto in buonissimo stato. Fabrizio Lavezzari - Via Resegone, 7 - 20051 Limbiate (Milano) - Tel. 02/9962845.

Vendo per micro NE schede: LX392 (completo di 32 K) a L. 80.000 - LX 385 a L. 80.000 - LX 382 (CPU) a L. 100.000. Tutte collaudate e funzionanti. Tratto zona Treviso e provincia. Renato Severin - Via Casaria, 23 - 31030 Biancade (TV).

Attenzione chi possiede una programmabile o un calcolatore e risiede nel Molise (possibilmente Campobasso) è pregato di scrivermi per creazione - Centro scambio e promozione software Molise. Allegare bollo. Vincenzo Carrone - Via Pascoli, 67 - 86100 Campobasso - Tel. 0874/91995.

Vendo personal computer Genie 1 16 K RAM - 12 K ROM, registratore incorporato, completo di manuali tecnici, riviste, editor-assembler, livello 3 Basic e moltissimi programmi. Tutto in perfetto stato a L. 850.000. Enrico Lago - C.so Belgio, 132 - 10153 Torino - Tel. 011/8990260.

PL/I-80, display manager, BT-80 *compro/scambio* con altri linguaggi e/o programmi per CP/M (Cobol, Fortran, Ingegneria, ecc.). Scrivere o telefonare a: Mario Guglielmetti - Via Milano, 19 - 38100 Trento - Tel. 913476.

HP431C perfetta + modulo standard + modulo statistica *vendo* a L. 300.000 purchè entro dicembre '83. Valore commerciale L. 455.000 circa. Spedizione contrassegno immediata. Francesco Lentini - Via Aschenez prolungam. 2/M - 89100 - Reggio Calabria - Tel. 0965/29257.

UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 3 N. 15 FEBBRAIO 1984

DIRETTORE RESPONSABILE: Giampietro Zanga

DIRETTORE: Pietro Dell'Orco

REDAZIONE: Lorenzo Barrile

COORDINAMENTO: Riccardo Paolillo

CONSULENZA: Giacomo Bortone

HANNO COLLABORATO A QUESTO NUMERO:
B. Del Medico, U.G. Barzaghi, A. Guida, I. Parbuono, G. Morpurgo, A. Stecchina, M. Spero, M. Lenzi, S. Borsani, M. Morchio.

GRAFICA E IMPAGINAZIONE: Luigi Chiesa

PUBBLICITA': Concessionario per l'Italia e l'Estero
Reina S.r.l. Via Washington, 50 - 20146 Milano
Tel. (02) 4988066/7/8/9/060 (5 linee r.a.)
Telex 316213 REINA I

FOTOCOMPOSIZIONE: LINEACOMP S.r.l.
Via Rosellini, 12 - 20124 Milano

STAMPA: REWEBA - Brescia

Concessionario esclusivo per la diffusione in Italia e all'Estero:
SODIP - Via Zuretti, 25 - 20125 Milano

AUTORIZZAZIONE ALLA PUBBLICAZIONE:
Tribunale di Milano n. 69 del 20/2/1982

Spedizione in abbonamento postale Gruppo III/70
Prezzo della rivista L. 3.500. Numero arretrato L. 7.000.
Abbonamento annuo (10 numeri) L. 28.000; per l'Estero L. 44.800

I versamenti vanno indirizzati a: Gruppo Editoriale Jackson
Via Rosellini, 12 - 20124 Milano - mediante emissione di assegno bancario, cartolina vaglia o utilizzando il C/C postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

Il Gruppo Editoriale Jackson è iscritto nel registro Nazionale della stampa al n. 117 - Vol. 2 - Foglio 129 in data 17-8-1982

GRUPPO EDITORIALE JACKSON S.r.l.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE:
Via Rosellini, 12 - 20124 Milano - Telefoni: 68.80.951/2/3/4/5

SEDE LEGALE: Via Vincenzo Monti, 15 - 20123 Milano

DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina

COORDINAMENTO EDITORIALE: Daniele Comboni

DIREZIONE AMMINISTRATIVA: Giuliano Di Chiano

CEDOLA DI COMMISSIONE LIBRARIA PERSONAL SOFTWARE

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. Città Provincia

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Pagherò al postino il prezzo indicato + L. 2.000 per contributo fisso spese di spedizione.

□ Allego assegno n° .... di L. ....

□ Non abbonato □ Abbonato sconto 20% □ l'Elettronica □ Elettronica Oggi □ Automazione Oggi □ Elektor
□ Informatica Oggi □ Computerworld □ Bit □ Personal Software □ Strumenti Musicali □ Videogiochi

Data .... Firma ....

PICCOLI ANNUNCI PERSONAL SOFTWARE

Sei un lettore di PERSONAL-SOFTWARE e vuoi entrare in contatto con tutti gli altri lettori per comperare, cambiare o vendere software? Spedisci questo tagliando a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome .... Cognome ....

Via .... C.A.P. ....

Città .... Tel. ....

Esposizioni Internazionali dell'Automazione

...1982 Parigi "MESUCORA"... 1983 Düsseldorf "INTERKAMA"

1984 MILANO - B.I.A.S.

Solo il BIAS nel 1984 in Europa presenta l'Automazione e la Microelettronica



studio martinetti

19° Convegno Mostra Internazionale dell'Automazione Strumentazione e Microelettronica

Fiera di Milano
29 novembre - 4 dicembre 1984

- Sistemi e Strumentazione per l'Automazione la regolazione ed il controllo dei processi Robotica, sensori e rilevatori
- Apparecchiature e Strumentazione per laboratorio, collaudo e produzione
- Componentistica, sottoassiemi periferiche ed unità di elaborazione
- Micro, Personal Computer, Software e accessori

E.I.O.M. Ente Italiano Organizzazione Mostre
Segreteria della Mostra
Viale Premuda 2
20129 Milano
tel. (02) 796096/421/635 - telex 334022 CONSEL

in concomitanza con la 8ª RICH e MAC '84